Distributed Document Clustering and Cluster Summarization in Peer-to-Peer Environments

by

Khaled M. Hammouda

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada 2007

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis addresses difficult challenges in distributed document clustering and cluster summarization. Mining large document collections poses many challenges, one of which is the extraction of topics or summaries from documents for the purpose of interpretation of clustering results. Another important challenge, which is caused by new trends in distributed repositories and peer-to-peer computing, is that document data is becoming more distributed.

We introduce a solution for interpreting document clusters using keyphrase extraction from multiple documents simultaneously. We also introduce two solutions for the problem of distributed document clustering in peer-to-peer environments, each satisfying a different goal: maximizing local clustering quality through collaboration, and maximizing global clustering quality through cooperation.

The keyphrase extraction algorithm efficiently extracts and scores candidate keyphrases from a document cluster. The algorithm is called *CorePhrase* and is based on modeling document collections as a graph upon which we can leverage graph mining to extract frequent and significant phrases, which are used to label the clusters. Results show that CorePhrase can extract keyphrases relevant to documents in a cluster with very high accuracy. Although this algorithm can be used to summarize centralized clusters, it is specifically employed within distributed clustering to both boost distributed clustering accuracy, and to provide summaries for distributed clusters.

The first method for distributed document clustering is called *collaborative peer-to-peer document clustering*, which models nodes in a peer-to-peer network as collaborative nodes with the goal of improving the quality of individual local clustering solutions. This is achieved through the exchange of local cluster summaries between peers, followed by recommendation of documents to be merged into remote clusters. Results on large sets

of distributed document collections show that: (i) such collaboration technique achieves significant improvement in the final clustering of individual nodes; (ii) networks with larger number of nodes generally achieve greater improvements in clustering after collaboration relative to the initial clustering before collaboration, while on the other hand they tend to achieve lower absolute clustering quality than networks with fewer number of nodes; and (iii) as more overlap of the data is introduced across the nodes, collaboration tends to have little effect on improving clustering quality.

The second method for distributed document clustering is called *hierarchically-distributed document clustering*. Unlike the collaborative model, this model aims at producing one clustering solution across the whole network. It specifically addresses scalability of network size, and consequently the distributed clustering complexity, by modeling the distributed clustering problem as a hierarchy of *node neighborhoods*. Summarization of the global distributed clusters is achieved through a distributed version of the CorePhrase algorithm. Results on large document sets show that: (i) distributed clustering accuracy is not affected by increasing the number of nodes for networks of single level; (ii) we can achieve decent speedup by making the hierarchy taller, but on the expense of clustering quality which degrades as we go up the hierarchy; (iii) in networks that grow arbitrarily, data gets more fragmented across neighborhoods causing poor centroid generation, thus suggesting we should not increase the number of nodes in the network beyond a certain level without increasing the data set size; and (iv) distributed cluster summarization can produce accurate summaries similar to those produced by centralized summarization.

The proposed algorithms offer high degree of flexibility, scalability, and interpretability of large distributed document collections. Achieving the same results using current methodologies require centralization of the data first, which is sometimes not feasible.

Acknowledgements

This thesis would not be possible without the support of many individuals, to whom I would like to express my gratitude. I will always be indebted to my supervisor, Prof. Mohamed Kamel, for his key role in my development as a person and as a researcher. He offered me support, encouragement, guidance, and most importantly trust. His input and guidance was invaluable to the quality and contribution of the work presented in this thesis, as well as in other publications. His trust and support was instrumental in giving me confidence to achieve many accomplishments.

I would like also to thank many faculty members of the University of Waterloo, most notably my committee members, Prof. Otman Basir, Prof. Robin Cohen, and Adj. Prof. Yang Wang, for their valuable input and suggestions. I would like also to thank Prof. Fakhri Karray and Prof. Hamid Tizhoosh for valuable discussions and ideas.

I wish to thank many of my colleagues at the Pattern Analysis and Machine Intelligence (PAMI) Lab, especially Shady Shehata, Masoud Makrehchi, Hanan Ayad, and Rasha Kashef, for valuable discussions and feedback. I would like also to thank former members of the PAMI group, especially Dr. Nayer Wanas, Dr. Jan Bakus, Dr. Khaled Shaban, and Dr. Ossama El Badawy, for many useful discussions and insights. I would like also to thank Diego Matute, a CS graduate, for helping me with part of the keyphrase scoring work. I would like to thank the PAMI administrative secretaries, Sue Havitz, Margaret Ulbrick, and Heidi Campbell, and the Systems Design graduate coordinator Vicky Lawrence for their help and support during my graduate studies. I would like to thank NSERC and the LORNET network for their financial support.

I would like to thank Hazem Shehata, Ahmed Youssef, Mohamed El-Abd, Ismael El-Samahy, Hassan Hassan, Ayman Ismail, Muhammad Nummer, Mohamed El-Dery, Hatem Zeineldin, Mohamed Elsaid, Mohamed El-Gebaly, and Mohamed Kamal, for being such

great friends.

Finally I would like to thank my wife, Shaimaa, for her unconditional support and love, without which many things would not be possible. I would like also thank my mother Farida, my late father Mahmoud, and my sisters Ghada and Dalia, for their support and encouragement throughout my life.

Contents

List of Tables

List of Figures

List of Algorithms

CHAPTER 1

Introduction

This thesis embodies research that aims at advancing the state of the art in distributed text mining, specifically distributed document clustering and cluster summarization. Document clustering is regarded as a key technology for intelligent unsupervised categorization of content in text form of any kind; *e.g.* news articles, web pages, learning objects, electronic books, even textual metadata. Information retrieval, provides access to information. Categorization, on the other hand, provides *organized*, *summarized*, *browsing-oriented* access to information, and in this regard it intelligently complements basic information retrieval.

Information environments today possess two key characteristics that sets them apart from information environments decades ago: (a) information is no longer maintained in central databases (*e.g.* the Web); and (b) computers processing this information are no longer central supercomputers, but rather a huge network of computers of all scales.

The problem of document clustering thus becomes more complex under those circumstances. How can documents distributed across a large number of sites be clustered together

in an efficient way? And can we interpret the results of such distributed clustering? The work presented in this thesis answers those two questions.

First, a novel approach for document cluster summarization is proposed. The approach utilizes a robust document phrase-based model to efficiently extract keyphrases from multiple documents simultaneously, and uses those keyphrases to label a cluster of documents. This provides a cluster interpretation capability for all sorts of document clustering methods, either centralized or distributed. As an enabling method for distributed clustering, keyphrase extraction is utilized both in boosting distributed clustering accuracy, and for providing summaries of distributed clusters.

Second, two novel approaches for distributed document clustering are proposed. The first is based on collaborative clustering between a set of nodes, and utilizes the cluster summarization technique mentioned above to produce compact cluster representation to be exchanged between peers during collaboration. This approach aims at producing locally-optimized document clusters for each node, thus final clusters are unique at each node.

The second distributed document clustering approach aims at scalability by structuring peer-to-peer networks as a hierarchy of node neighborhoods. Clustering is performed within neighborhood boundaries and then combined up through the hierarchy. This approach aims at producing globally-optimized document clusters for the entire network, thus final clusters are identical at each node. Summarization of the distributed clusters is performed through distributed keyphrase extraction by a distributed variant of the original algorithm.

This introduction is organized as follows. Section 1.1 gives an overview of data clustering, document clustering, and distributed clustering. Section 1.2 introduces the cluster summarization approach. Section 1.3 introduces the two distributed document clustering approaches. Section 1.4 lists the contributions made in this thesis. Finally, section 1.5 provides an overview of the thesis organization.

1.1 Overview

Analysis of data can reveal interesting, and sometimes important, structures or trends in the data that reflect a natural phenomenon. Discovering regularities in data can be used to gain insight, interpret certain phenomena, and ultimately make appropriate decisions in various situations. Finding such inherent but invisible regularities in data is the main subject of research in *Data Mining*.

1.1.1 Data clustering

One type of regularity in data is the natural grouping of objects into clusters. Data clustering is a data mining technique that enables the abstraction of large amounts of data by forming meaningful groups or categories of objects, formally known as *clusters*, such that objects in the same cluster are similar, and those in different clusters are dissimilar.

A cluster of objects indicates a level of similarity between those objects such that we can consider them to be in the same category, thus simplifying our reasoning about them considerably. For example, we can consider computers with one processor and limited memory to be in the category of *personal computers*, while those with multiple processors and large memory are in the category of *server computers*, without having to refer to every computer instance in those categories. Consequently, we can characterize a group of objects by referring to the common features that differentiate them from other groups.

The choice of which objects belong to the same cluster depends on the clustering model. In *distance-based clustering* the decision is based on the distances between objects, and thus requires definition of a distance (or inversely a similarity) measure defined over the object feature space. In *conceptual clustering* there is a common concept (a statistical model that emphasizes common features in a cluster) that ties objects in the same cluster,

and the decision of including objects into a cluster is based on how well the features of an object fit that concept.

Types of clustering algorithms

Clustering algorithms fall into a number of categories depending on their various aspects.

- *Hard clustering*, *e.g.* k-means, assigns each object exclusively to one cluster, thus creating a disjoint set of clusters. *Probabilistic*, *e.g.* expectation-maximization, and *fuzzy clustering*, *e.g.* fuzzy c-means, assigns for each object a degree of membership to each cluster, thus creating overlapping clusters.

- *Hierarchical clustering*, *e.g.* hierarchical agglomerative clustering, creates a *dendogram* of clusters such that clusters can contain sub-clusters. It works either bottom-up by merging clusters into larger clusters on the next level of the hierarchy, or top-down by splitting clusters into sub-clusters. *Flat clustering*, on the other hand, produces a flat set of clusters with no ordering or subsumption between them.

- *Density-based clustering*, *e.g.* DBSCAN [33], forms clusters by finding density-connected regions in the feature space.

- *Neural network-based clustering*, *e.g.* SOM [68], utilizes a neural network approach that automatically tunes the network weights such that similar objects tend to be close to each other.

Applications of clustering

Clustering is used in a wide range of applications, such as marketing, biology, psychology, astronomy, image processing, and text mining. For example, in marketing it is used to find

groups of customers that share common behavior for the purpose of market segmentation and targeted advertisement. In biology it is used to form a taxonomy of species based on their features. In image processing it is used to segment texture in images to differentiate between various regions or objects. Clustering is also practically used in many statistical analysis software packages for general purpose data analysis.

1.1.2 Document clustering

Although clustering can be applied to many types of data, the focus of this thesis is on clustering text documents, a field known in the literature as *document clustering* [94] which is a subfield of *text mining*. Document clustering deals with the unsupervised partitioning of a document collection into meaningful groups based on their textual content, usually for the purpose of topic categorization; *i.e.* documents in one cluster belong to a certain topic, while different clusters represent different topics. Unlike document classification – which is a supervised learning method that requires prior knowledge of document categories to train a classifier, document clustering is an unsupervised learning method that does not rely on prior categorization knowledge.

Document clustering has many applications, such as clustering of search engine results to present organized and understandable results to the user (*e.g.* Vivisimo[1]), clustering documents in a collection (*e.g.* digital libraries), automated (or semi-automated) creation of document taxonomies (*e.g.* Yahoo! and Open Directory styles), and efficient information retrieval by focusing on relevant subsets (clusters) rather than whole collections. News aggregation is becoming a common application of document clustering, exemplified by the Google News[2] service, which uses document clustering to group news articles from multiple

[1] `www.vivisimo.com`
[2] `news.google.com`

news sources, providing an automated compilation of recent news.

1.1.3 Challenges in clustering

There is a number of problems associated with clustering, which are outlined here:

- choice of a good (dis)similarity measure,
- choice of the number of clusters,
- ability to perform incremental update of clusters without re-clustering,
- properly dealing with outliers,
- interpretation of clustering results,
- tackling distributed data,
- scalability, both in terms of the number of objects and the number of dimensions, and
- evaluation of clustering quality,

In this thesis, three of those challenges are addressed: interpretation of clustering results, scalability, and tackling distributed data. Interpreting clustering results is addressed through document cluster summarization using a novel keyphrase extraction algorithm, while scalability and tackling distributed data are addressed through novel distributed clustering algorithms. The cluster summarization algorithm plays a key role both as a stand-alone algorithm for post-processing clustering results, as well as in facilitating the distributed clustering algorithms. Before introducing those algorithms, an overview of distributed clustering is in order.

1.1.4 Distributed clustering

Han and Kamber [49] describe the need for parallel and distributed mining algorithms:

> "The huge size of many databases, the wide distribution of data, and the computational complexity of some data mining methods are factors motivating the development of **parallel and distributed data mining algorithms**. Such algorithms divide the data into partitions, which are processed in parallel. The results from the partitions are then merged."

With the continuous growth of data in distributed networks, it is becoming increasingly important to perform clustering of distributed data in-place, without the need to pool it first into a central location. We introduce a general view of how clustering can be applied in distributed environments, which is outlined in Table 1.1.

Table 1.1: Types of data and clustering process distribution

	Centralized Data	Distributed Data
Centralized Clustering	CD-CC	DD-CC
Distributed Clustering	CD-DC	DD-DC

Centralized Data - Centralized Clustering (CD-CC) This is the standard approach where the clustering process and data both reside on the same machine. Other distributed models can be mapped to CD-CC by pooling the distributed data into one location and performing centralized clustering on it.

Distributed Data - Centralized Clustering (DD-CC) Data is dispersed across a number of nodes, while the clustering process runs on a single machine. Web mining is an example of DD-CC, where a single machine crawls and mines web pages from a large number of remote web sites.

Centralized Data - Distributed Clustering (CD-DC) Data is stored in one location, while clustering processes run on different machines accessing the same data. This is a typical case of *parallel processing*, such as in compute clusters and grid computing.

Distributed Data - Distributed Clustering (DD-DC) The highest level of distribution, where both the data and the clustering process are distributed.

In general, centralized clustering usually implies high computational complexity, while distributed clustering usually aims for speedup but suffers from communication overhead. The general goal of distributed clustering is achieving a level of speedup that outweighs communication overhead. The distributed clustering algorithms presented in this thesis fall under the DD-DC category.

Adopting distributed clustering introduces another factor affecting algorithm scalability: number of nodes. Data privacy is also an issue in distributed clustering, since the participating sites may not be willing to share their data with peers.

In general, there are two architectures in distributed clustering: peer-to-peer and facilitator-worker.

- In the peer-to-peer model all nodes perform the same task and exchange the necessary information to perform their clustering goals.
- In the facilitator-work model one node is designated as a facilitator, and all others are considered worker nodes. The facilitator is responsible for partitioning the task among the workers and aggregating their partial results.

The goal of distributed clustering can be either to produce globally or locally optimized clusters. Globally optimized clusters reflect the grouping of data across all nodes, as if data

from all nodes were pooled into a central location for centralized clustering. As a result, at the end all nodes acquire the same clustering solution, but local data stays the same. Globally optimized clustering is suitable for speeding up clustering of large data sets by partitioning the data among many nodes. Both the peer-to-peer [24, 25, 23] and the facilitator- worker [62, 64, 67] models can be used to achieve globally optimized clustering.

On the other hand, locally optimized clusters create a different set of clusters at each node, taking into consideration remote clustering information and data at other nodes. This implies exchange of data between nodes so that certain clusters appear only at specific nodes. Locally optimized clusters are useful when whole clusters are desired to be in one place rather than fragmented across many nodes. It is also only appropriate when data privacy is not a big concern. Both the peer-to-peer model [47] and the facilitator-worker [76] models can be used to achieve locally optimized clustering.

1.2 Summarization of Document Clusters

While document clustering can be valuable for categorizing documents into meaningful groups, the usefulness of categorization cannot be fully appreciated without labeling those clusters with the relevant keywords or keyphrases that describe the various topics associated with them. A highly accurate keyphrase extraction algorithm, called CorePhrase [44], is proposed for this particular purpose.

CorePhrase works by building a complete list of phrases shared by at least two documents in a cluster. Phrases are assigned scores according to a set of features calculated from the matching process. The candidate phrases are then ranked in descending order and the top L phrases are output as a label for the cluster.

While this algorithm on its own is useful for labeling document clusters, it is used to

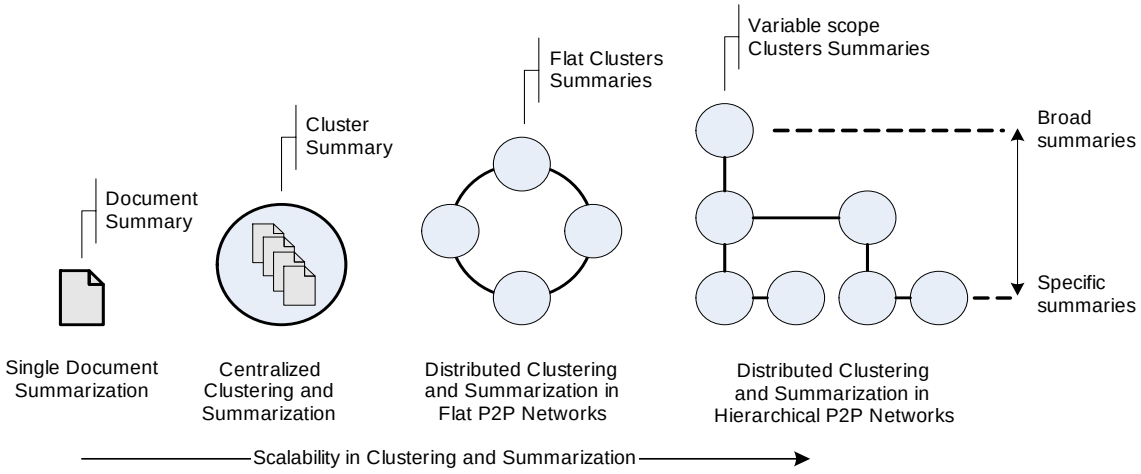


Figure 1.1: Levels of Clustering and Summarization

produce cluster summaries for the collaborative clustering algorithm (described below). Cluster keyphrase summaries are exactly what is used to succinctly inform remote nodes of the content of local document clusters, which in turn is used to judge the similarity between remote data and local clusters.

A distributed version of this algorithm is also used in the hierarchically-distributed clustering algorithm (described below) to produce summaries for the globally distributed clusters.

Figure 1.1 illustrates the various levels at which summarization of document clusters can take place.

- Keyphrase extraction can be applied to a single document for labeling the document; this is mainly used in generating metadata (e.g. title, description, keywords) that can be associated with the document.

- A centralized document cluster can be summarized and labeled using keyphrase extraction.

- Distributed document clusters in a flat peer-to-peer network can be summarized. Cluster summaries can be exchanged between peers to facilitate collaborative clustering.

- Distributed document clusters in a hierarchical peer-to-peer network can be summarized. Cluster summaries can be accessed at different levels of the hierarchy, thus providing variable scope of summaries ranging from specific to broad.

Evaluation of the accuracy of CorePhrase shows that it can accurately extract those phrases that match the manually labeled topic of clusters, and is able to rank those matching phrases in the top two or three keyphrases.

1.3 Distributed Document Clustering

Both locally-optimized and globally-optimized distributed clustering are addressed in this thesis. A concept of collaborative peer-to-peer nodes is adopted to solve both problems, with a focus on application to document clustering. Locally-optimized distributed clustering is discussed in chapter 4. Part of the algorithm is dependent on keyphrase extraction from document clusters, which is presented first in chapter 3. The globally-optimized distributed clustering, which is based on a hierarchically-distributed peer-to-peer architecture, is discussed in chapter 5.

Notations and Definitions

A distributed network of N_P peer nodes is modeled as a set $\mathbf{P} = \{p_i\}_{i=1}^{N_P}$. Unless otherwise noted, any node can communicate with any other node for the purpose of exchange of data or control messages.

The global data set is represented as $\mathbf{D} = \{d_i\}_{i=1}^{N_D}$. Each node p_i stores locally a subset of the global data set $\mathbf{D}_i \subseteq \mathbf{D}$. Local data sets may overlap; *i.e.* $\forall i, j \neq i : \mathbf{D}_i \cap \mathbf{D}_j$ is not necessarily empty. The global data set is partitioned among network nodes using a partitioning process $\mathcal{P}$ that chooses a uniformly-distributed random subset of $\mathbf{D}$ for each node, subject to partitioning factor α that determines how much overlap of data is required between nodes:

$$\{\mathbf{D}_i\} = \mathcal{P}(\mathbf{D}, \alpha), \quad \frac{1}{N_P} \leq \alpha \leq 1 \tag{1.1}$$

Thus $|\mathbf{D}_i| = N_{D_i} = \alpha \cdot N_D$.

A clustering solution is a set of clusters $C = \{c_k\}_{k=1}^{N_C}$ defined over a data set D, such that each cluster c_k contains a subset of D. In distance-based clustering algorithms, the input data D is transformed into a symmetric distance (or similarity) matrix S, over which the clustering algorithm operates, such as the case in hierarchical clustering and graph partitioning algorithms. In general, the clustering solution is acquired using a clustering algorithm $\mathcal{A}$:

$$C = \begin{cases} \mathcal{A}(D) & \text{if input is } D \\ \mathcal{A}(S) & \text{if input is } S \end{cases} \tag{1.2}$$

1.3.1 Locally-optimized Distributed Clustering

The objective of locally-optimized distributed clustering is to employ collaboration between nodes to improve the quality of local clustering solutions [47]. This implies that there is no common set of clusters across nodes, but rather local clusters that reflect the characteristics of local data sets. However, by strategically moving (or copying) certain data objects from one node to another, the quality of local clusters is boosted. This effectively creates a network where certain nodes have authority over certain clusters, which can simplify query-answering in P2P networks by routing queries to relevant nodes only, instead of

flooding the query throughout the network.

Collaboration between nodes is achieved through sharing of summarized local clustering information across the network. A node p_i starts by creating an initial local clustering C_i^I, and creates a summary for it, Λ_i, using keyphrase extraction from the document clusters (other types of summarization are possible depending on the type of data). The node then shares Λ_i with all other nodes. After node p_j receives cluster summary information Λ_i it computes a similarity matrix between its own data set D_j and Λ_i, and builds a recommendation list D_{ji}^+ of documents that are of high similarity to remote clusters (called peer-positive docuemnts) to be merged into remote clusters. When p_i receives D_i^+, it performs a merging step to include those documents that it believe will contribute positively to its own clusters.

More formally, let C_i^I be the initial clustering solution computed by node p_i over the local data set D_i. Then the final clustering solution after collaboration with all peers is:

$$C_i^F = \text{Merge}(C_i^I, \{D_{ji}^+\}_{j \neq i}) \tag{1.3a}$$

$$D_{ji}^+ = \text{FindSimilar}(\Lambda_i, D_j)|_T \tag{1.3b}$$

where *Merge* is the operation for merging recommendation lists D_{ji}^+ from remote nodes into the initial local clusters, and *FindSimilar* is the operation for determining the similarity between cluster summaries D_{ji}^+ and the data D_j at node p_j, subject to the similarity threshold T. Figure 1.2 illustrates this process showing only interaction between two nodes.

Both the initial clustering and the merging step are based on a similarity-histogram based clustering algorithm [41]. The algorithm represents the distribution of pairwise similarities in clusters as a histogram with an associated property called *Histogram Ratio*.

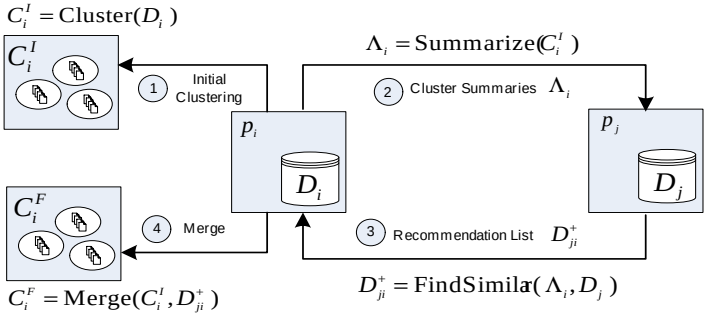


Figure 1.2: Abstract model of the locally-optimized collaborative clustering technique

The histogram ratio represents the fraction of similarities above a certain threshold to the total set of similarities. The higher the ratio the more coherent the cluster. Addition or merging of documents into clusters is controlled by keeping this ratio as high as possible.

Results on large sets of distributed document collections (see chapter 6) show that

- such collaboration technique achieves significant improvement in the final clustering of individual nodes;
- networks with larger number of nodes generally achieve greater improvements in clustering after collaboration relative to the initial clustering before collaboration, while on the other hand they tend to achieve lower absolute clustering quality than networks with fewer number of nodes; and
- as more overlap of the data is introduced across the nodes, collaboration tends to have little effect on improving clustering quality.

1.3.2 Globally-optimized Distributed Clustering

The goal of globally-optimized clustering is to compute one set of clusters over all local data sets, as if the data were pooled into one site and a centralized clustering algorithm was applied to it. Globally-optimized clustering algorithms can be either exact or approximate. Exact algorithms produce a set of clusters identical to ones produced by performed centralized clustering on the whole data. Approximate algorithms do not necessarily produce exact solutions, but usually produce ones that are close enough, with the added benefit of being less complex both in terms of computation and communication.

To address the problem of scalability and modularity of globally-distributed clustering, a hierarchically-distributed peer-to-peer clustering architecture and algorithm are proposed. Nodes in the network are partitioned into *neighborhoods*, the size of each is usually an order of magnitude lower than the size of the network. A distributed variant of the k-means algorithm is applied within each neighborhood to arrive at a set of centroids[3]. The algorithm works iteratively by updating local centroids after receiving centroid information from peers in the same neighborhood:

$$C_i^t = \text{Update}(\{C_j^{t-1}\}) \tag{1.4}$$

The problem is now reduced into combining the set of centroids from the various neighborhoods. One solution is to allocate a facilitator node that combines multiple sets of centroids, but this would defeat the scalability of the method in cases when the number of neighborhoods is very large.

To address this issue, a node in each neighborhood is designated as a supernode, and the neighborhood concept is recursively applied to all supernodes such that we create a

[3]A centroid is the mean of a set of document vectors.

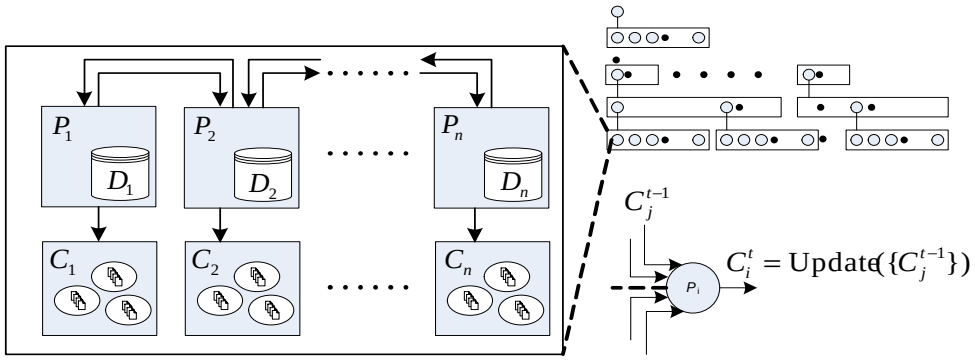


Figure 1.3: Abstract model of the globally-optimized hierarchically-distributed clustering technique

multi-layer hierarchical network. At each layer we combine the set of centroids computed from the lower layers, until we arrive at the root with a single set of centroids. This set is the globally-optimized clustering solution across all nodes. Figure 1.3 illustrates this process showing a global view of the node hierarchy, and a focus on one neighborhood in addition to the abstract process of iterative cluster update.

We define a relationship between the hierarchy height H, the neighborhood size S_Q, and the total number of nodes N_P. This architecture faces a tradeoff between speedup and the approximate clustering quality. The taller the hierarchy the less the quality of clustering, but the greater the speedup. Results on large document sets show that:

- we can achieve decent speedup by making the hierarchy taller, but at the expense of clustering quality which degrades as we go up the hierarchy due to working with centroid information only rather than actual data at higher levels; and

- we should not increase the number of nodes in the network beyond a certain level without increasing the data set size due to fine-grained data partitioning across nodes.

Finally, a method for generating keyphrase summaries for the distributed clusters based

on CorePhrase is proposed. At level 0 of the hierarchy, each node in a certain neighborhood produces a set of core keyphrases for each cluster. Then nodes exchange those keyphrases, intersecting the list with all other keyphrase summaries from their neighbors. The result of the intersection is called the cluster core and is kept for the next iteration by all nodes. At the same time, each node intersects its neighbors' keyphrases with its local data to produce a list of keyphrases to be used in the next iteration. This process repeats until the desired number of keyphrases is extracted, or the intersection yields an empty set.

At higher levels, when a set of cluster centroids are merged, their respective keyphrase summaries are merged as well. At the root of the hierarchy, one set of summary keyphrases for each cluster is obtained.

1.4 Contributions

The following is a list of contributions in this thesis.

Document Clustering and Summarization

- Development of a keyphrase extraction algorithm from document clusters. The algorithm efficiently finds all matching phrases between documents in a cluster, and subsequently ranks them to discover the highly relevant keyphrases for the cluster. Higher accuracy is demonstrated over keyword-centroid algorithms.

- Combining the keyphrase extraction algorithm with the collaborative clustering algorithm to provide collaborative document clustering capability. The extracted keyphrases are used as cluster summaries that are shared between nodes in the network for the purpose of identifying similar documents in remote peers.

- Combining the keyphrase extraction algorithm with the hierarchically-distributed clustering algorithm to provide summaries for the globally distributed clusters.

Distributed Clustering

- Development of a collaborative peer-to-peer clustering algorithm. The algorithm provides locally-optimized distributed clustering through the exchange of cluster summaries between peers, upon which recommendation of merging remote data into local clusters is based. Final clusters after collaboration show significant improvement in quality over initial clusters before collaboration.

- Development of a hierarchically-distributed peer-to-peer architecture and algorithm. The architecture aims for scalability through modularizing the network into a hierarchy of peer neighborhoods. The clustering algorithm directly exploits the architecture through recursively computing cluster centroids within neighborhoods, and merging multiple centroid solutions up the hierarchy. The algorithm scales with a large number of nodes, with clustering quality comparable to the centralized version.

Data Clustering

- Definition of a new clustering algorithm based on positive and negative contribution of specific data objects to clusters. The algorithm maximizes the quality of clusters by removing negatively contributing objects while incorporating positively contributing objects. The method is guided by a histogram representation of the pairwise similarity distribution in clusters.

These contributions provide necessary methods for distributed document repositories, such as digital libraries or distributed knowledge bases, to collaborate with each other to

discover regularities and patterns that may span multiple sites.

1.5 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 provides a background and review of the research subjects related to the work herein. Chapters 3, 4, and 5 introduce the main work in the thesis: chapter 3 introduces the text mining algorithm for keyphrase extraction from document clusters; chapter 4 introduces the distributed collaborative peer-to-peer document clustering method; and chapter 5 introduces the hierarchically-distributed peer-to-peer clustering method. Experimental setup and results and given in chapter 6. Finally, a summary, conclusions, and future research are presented in chapter 7.

CHAPTER 2

Background and Related Work

Data mining is a broad field that is at the cross-roads of many disciplines. In this chapter we will review the relevant literature as it relates to distributed document clustering. Figure 2.1 illustrates the various fields of study which the work in this thesis stems from or extends, with an emphasis on positioning the work within those fields. Not all (sub)fields were shown; only those important enough or directly relevant to the work are included.

The highlighted areas in the figure indicate the research areas directly touched by this research, namely: **Distributed Clustering**, Peer-to-Peer Clustering - specifically **P2P Document Clustering**, and Text Mining - specifically **Document Clustering**. As can be seen, the work is based largely on applying text data mining (document clustering) within distributed computing environments (peer-to-peer networks).

The organization of this literature review can be categorized into the following broad areas: (1) Text Mining, (2) Clustering Algorithms, (3) Distributed Clustering, and (4) Agent-based Data Mining. There are many ways to break down the literature on text and dis-

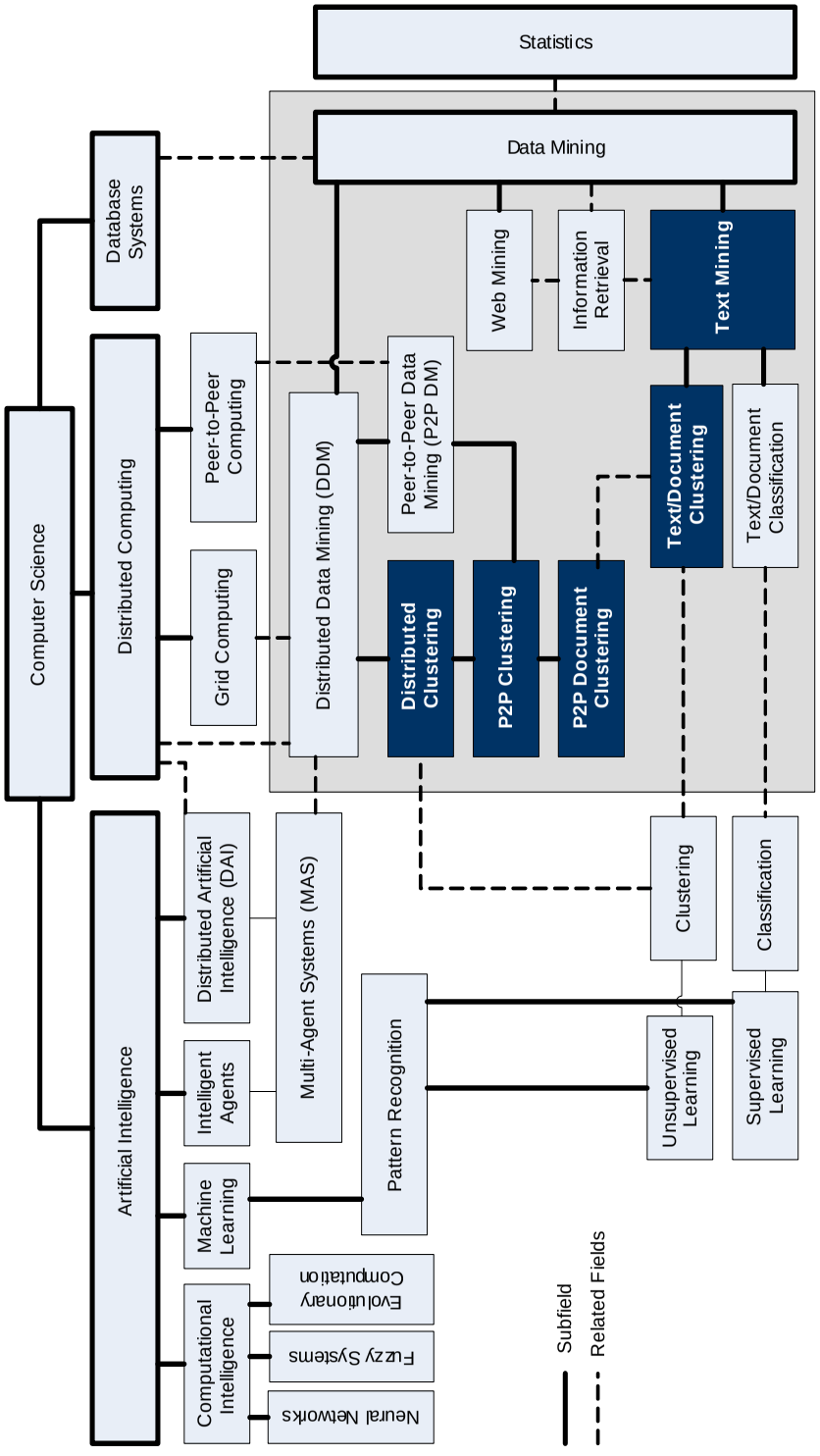


Figure 2.1: Fields of study relevant to Distributed Clustering

tributed data mining. Regardless of how we choose to look at it, we should be able to appreciate the different approaches that have been reported in the literature by the end of this chapter.

The first two sections on text mining and clustering algorithms are largely based on centralized approaches, and thus those sections discuss basic centralized models and algorithms only. The latter two sections focus on distributed architectures and algorithms for performing the equivalent of the centralized approaches in distributed environments.

2.1 Text Mining

The term *text mining* was first proposed by Feldman and Dagan in [35]. According to a survey by Kosala and Blockeel on Web mining [69], currently the term text mining has been used to describe different applications such as text categorization [54, 100, 104], text clustering [107, 110, 12, 70, 100], empirical computational linguistic tasks [52], exploratory data analysis [52], finding patterns in text databases [35, 36], finding sequential patterns in text [74, 2, 3], and association discovery [87, 100].

Text Mining vs. Information Retrieval

Text Mining is a field that is considered as an extension of *Data Mining* in general, also known as *Knowledge Discovery in Databases* [34]. It is a term that sometimes gets confused with *Information Retrieval*, which is a different, but related, field. Mining is not Retrieval. The goal of information retrieval (or information access) is to help users find documents that satisfy their needs [4]. The problem is not that the information is not known, but rather that the information coexists with other valid pieces of information, and we just need to "home in" on this information. This search-centric view misses the point that we

might actually want to treat the information as a large knowledge base from which we can extract new, never-before encountered information [20]. On the other hand, the goal of data mining is to discover new information from data, finding patterns across data sets, and separate signal from noise [52].

Document Categorization

It is also sometimes confusing when people refer to document categorization as data mining. Document categorization is a process of mapping the content of a document into one (or more) of a set of pre-defined labels. Although it is considered a machine learning task, it does not lead to discovery of new information; rather, it produces a classification of something already known. This is not to undermine document categorization; it has its own benefits in assigning documents to their respective categories automatically, thus relieving a human from such a daunting task. The argument here is just that it does not provide more than what is already known. Aas *et al*[1] give a good account of the text categorization literature, exposing the different types of text representation and classification methods.

Document Clustering

Clustering is a technique to group similar objects together, based on their (dis)similarity, to form a grouping of objects such that objects in the same group are most similar, while objects in different groups are most dissimilar [57]. In the context of text mining, clustering is a really powerful method for discovering interesting (inherent) grouping of documents, maybe to form a computer-aided information hierarchy, such as a Yahoo-like topic directory. A potential benefit is to let the documents categorize themselves. It might be possible to come up with groups of things that are recognized, but it might not be clear that they could be made into a category in advance. After clustering analysis is done, clusters could

be refined, and as new documents are introduced they can be automatically assigned using automatic classification.

Clusters are not the same as categories. Clusters are based on similarities found by the computerized analysis of the documents in the collection. Categories are pre-assigned groupings designed to be meaningful (and helpful) to people. Because categories are pre-assigned without reference to the content of the document collection it is likely that some categories will have many documents while other categories are empty [77].

2.1.1 Text Representation Models

In data mining in general, usually there is a fixed model for data that is assumed by most mining algorithms. This data model varies depending on the nature of the data itself. For problems that have numerical data, a straightforward numerical representation is assumed by the algorithms. However, in text mining we have free unstructured text data, which poses a problem of representation. An overview of the most common document representation models used in text mining is given here.

Document Data Models

Most text mining methods use the **Vector Space Model**, introduced by Salton in 1975 [92], to represent document objects. Each document is represented by a vector $\mathbf{d}$, in the term space, $\mathbf{d} = \{tf_1, tf_2, \ldots, tf_n\}$, where tf_i, $i = 1, \ldots, n$ is the term frequency in the document, or the number of occurrences of the term t_i in a document. To represent every document with the same set of terms, we have to extract all the terms found in the documents and use them as our feature vector[1]. Sometimes another method is used which

[1]Obviously the dimensionality of the feature vector is always very high, in the range of hundreds and sometimes thousands.

combines the term frequency with the inverse document frequency (TF-IDF) [92, 1]. The document frequency df_i is the number of documents in a collection of N documents in which the term t_i occurs. A typical inverse document frequency (idf) factor of this type is given by $\log(N/df_i)$. The weight of a term t_i in a document is given by:

$$w_i = tf_i \times \log(N/df_i). \tag{2.1}$$

To keep the dimension of the feature vector reasonable, only a small number of n terms with the highest weights in all the documents are chosen. Wong and Fu [107] showed that they could reduce the number of representative terms by choosing only the terms that have sufficient *coverage*[2] over the document set.

Some algorithms [61, 107] refrain from using term frequencies (or term weights) by adopting a binary feature vector, where each term weight is either 1 or 0, depending on whether it is present in the document or not. Wong and Fu [107] argued that the average term frequency in Web documents is below 2 (based on statistical experiments), which does not indicate the actual importance of the term, thus a binary weighting scheme would be more suitable to this problem domain.

Another model for document representation is called **N-gram** [98]. The N-gram model assumes that the document is a sequence of characters. Using a sliding window of size n, the original character sequence is scanned to produce all n-character sub-sequences. The N-gram approach is tolerant of minor spelling errors because of the redundancy introduced in the resulting n-grams. The model also achieves minor language independence when used with a stemming algorithm, which reduces inflected words to their stem form. Similarity in this approach is based on the number of shared n-grams between two documents.

Another model proposed by Zamir and Etzioni [109] is a phrase-based model based on

[2]The *Coverage* of a feature is defined as the percentage of documents containing that feature.

Suffix Trees. The model finds common phrase suffixes between documents and builds a suffix tree where each node represents part of a phrase (a suffix node) and associated with it are the documents containing this phrase-suffix. The approach clearly captures the information of word proximity, which is thought to be valuable for finding similar documents. However, the branching factor of this tree is questionably huge, especially at the first level of the tree, where every possible suffix found in the document set branches out of the root node. The tree also suffers a great degree of redundancy of suffixes repeating all over the tree in different nodes.

Finally, another phrase-based approach was proposed by Hammouda and Kamel [41] to facilitate matching phrases efficiently between documents. The model is called the **Document Index Graph** (DIG). It is a graph-based model in which nodes represent unique words along with term frequency information, and edges represent sequences of words. Since this model is used as the underlying representation model in the keyphrase extraction algorithm in chapter 3, as well as for the cluster summarization part of chapter 4, a brief definition of the DIG model is given here.

The DIG is a directed graph (digraph) $G = (V, E)$

where V : is a set of *nodes* $\{v_1, v_2, \ldots, v_n\}$, where each node v represents a unique word in the entire document set; and

E : is a set of *edges* $\{e_1, e_2, \ldots, e_m\}$, such that each edge e is an ordered pair of nodes (v_i, v_j). An edge from v_i to v_j indicates that the word v_j appears successive to the word v_i in some document.

Each document $\mathbf{d}_i$ is mapped to a document sub-graph g_i that represents the unique words and their sequences in that document (i.e. phrases). The DIG model is built incrementally by merging each document sub-graph into a cumulative graph that represents documents processed up to $\mathbf{d}_i$: $G_i = G_{i-1} \cup g_i$.

Upon merging a document sub-graph into the cumulative graph, it is possible to extract the matching phrases between the new document and all previous documents. The list of matching phrases between document $\mathbf{d}_i$ and $\mathbf{d}_j$ is computed by *intersecting* the subgraphs of both documents, g_i and g_j, respectively. Let $\mathbf{p}_{ij}$ denote such list, then:

$$\mathbf{p}_{ij} = g_i \cap g_j$$

A list of matching phrases between document $\mathbf{d}_i$ and all previously processed documents is computed by intersecting the document subgraph g_i with the cumulative DIG G_{i-1}. Let $\mathbf{p}_i$ denote such list, then:

$$\mathbf{p}_i = g_i \cap G_{i-1}$$

This process produces complete phrase matching output between every pair of documents in near-linear time, with arbitrary length phrases [43]. Thus, the model allows phrase-based similarity calculation, which was experimentally shown to be more accurate than single-word analysis.

Text Pre-processing

Before any feature extraction takes place, the document set is normally *cleaned* by removing stop-words [3] and then applying a stemming algorithm that converts different word forms into a similar canonical form. The most popular stemming algorithm is the Porter stemmer [90].

[3] Stop-words are very common words that have no significance for capturing relevant information about a document (such as "the", "and", "a", ... etc).

2.1.2 Document Similarity Measures

A key factor in the success of any clustering algorithm is the similarity measure adopted by the algorithm. In order to group similar data objects, a proximity metric has to be used to find which objects (or clusters) are similar. There are a large number of similarity metrics reported in the literature, only the most common ones are reviewed in this section.

The calculation of the (dis)similarity between two objects is achieved through some *distance* function, sometimes also referred to a *dissimilarity* function. Given two feature vectors $\mathbf{x}$ and $\mathbf{y}$ representing two objects it is required to find the degree of (dis)similarity between them.

A very common class of distance functions is known as the *family of Minkowski* distances [16], described as:

$$\|\mathbf{x}-\mathbf{y}\|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p} \tag{2.2}$$

where $\mathbf{x}, \mathbf{y} \in \Re^n$. This distance function actually describes an infinite number of the distances indexed by p, which assumes values greater than or equal to 1. Some of the common values of p and their respective distance functions are:

$$p = 1: \quad \text{Manhattan Distance} \quad \|\mathbf{x}-\mathbf{y}\|_1 = \sum_{i=1}^{n} |x_i - y_i| \tag{2.3}$$

$$p = 2: \quad \text{Euclidean Distance} \quad \|\mathbf{x}-\mathbf{y}\|_2 = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{2.4}$$

$$p = \infty: \quad \text{Tschebyshev distance} \quad \|\mathbf{x}-\mathbf{y}\|_\infty = \max_{i=1,2,\ldots,n} |x_i - y_i| \tag{2.5}$$

A more common similarity measure that is used specifically in document clustering is

the *cosine correlation* measure (used by [94, 21, 107]), defined as:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \tag{2.6}$$

where $(\cdot)$ indicates the vector dot product and $\|\cdot\|$ indicates the length of the vector. Strehl *et al* [95] provide a discussion on the impact of similarity measures on web page clustering.

Another commonly used similarity measure is the *Jaccard* measure (used by [70, 61, 51]), defined as:

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} \min(x_i, y_i)}{\sum_{i=1}^{n} \max(x_i, y_i)} \tag{2.7}$$

which in the case of binary feature vectors could be simplified to:

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{x} \cap \mathbf{y}|}{|\mathbf{x} \cup \mathbf{y}|} \tag{2.8}$$

It has to be noted that the term "distance" is not to be confused with the term "similarity". Those terms are opposite to each other in the sense of how similar the two objects are. Similarity *decreases* when distance *increases*. Another remark is that many algorithms employ the distance function (or similarity function) to calculate the similarity between two clusters, a cluster and an object, or two objects. Calculating the distance between clusters (or clusters and objects) requires a representative feature vector of that cluster (sometimes referred to as a *medoid*).

Some clustering algorithms make use of a *similarity matrix* . A similarity matrix is a $N \times N$ matrix recording the distance (or degree of similarity) between each pair of objects. Obviously the similarity matrix is a positive definite matrix so we only need to store the upper right (or lower left) portion of the matrix.

2.1.3 Keyphrase Extraction

An important task in text mining is given a set of text documents, extracting the essence of the information in those documents and presenting it in the form of the few keyphrases that capture the topic covered by those documents. One of the answers is *keyphrase extraction*, which is concerned with the analysis of text documents to extract highly relevant keyphrases from them.

Applications of keyphrase extraction are numerous. Turney [102] lists over a dozen applications that directly or indirectly depend on keyphrase extraction, to name a few: providing mini-summaries of large documents, highlighting keyphrases in text, text compression, constructing human-readable keyphrase-based indexes, interactive query refinement by suggesting improvements to queries, document clustering, and document classification.

Much of the work reported in the literature about keyphrase extraction has to do with single-document extraction, with roots in text summarization. There have been few attempts towards summarizing text clusters, or multi-document sets, which is one of the challenges addressed in this thesis. Moreover, those cluster summarization techniques use keyword-based frequency measures, with no use of keyphrases.

Two popular single-document keyphrase extraction algorithms are: *Extractor* by Turney [102], and *Kea* by Frank *et al* [37]. Extractor uses machine learning to extract keyphrases from individual documents, which employs a genetic algorithm to tune its parameters. Evaluation is based on the number of extracted keyphrases that match human generated keyphrases, which is claimed to achieve human-like performance, but could be biased towards the trained data. Kea is a single document summarizer, which employs a Bayesian supervised learning approach to build a model out of training data, then applies the model to unseen documents for keyphrase extraction.

Mana-Lopez *et al* [78] have applied multi-document summarization as a post-processing step of document clustering. Their approach is system-based with focus on the integration of the summarization with IR systems, and the differences between presentation of the retrieved results in the form of a pure ranked list, clustered lists, and cluster summaries.

Another system called *SNS* has been proposed by Radev and Fan [91] for summarizing search engine hit lists. It employs multi-document summarization using full sentences by ranking candidate sentences according to their relevance to the cluster centroid.

Among the methods using keyword-based cluster labeling is a one proposed by Neto *et al* [88], which uses Autoclass-based clustering with top frequent keywords for cluster labels. In addition to keyword-based cluster summarization, it is also used to perform single document summarization, using a variation of the popular tf×idf weighting scheme to score phrases. A similar approach for describing SOM-based text clusters using keywords has been proposed by Merkl and Rauber [82]. The method is called LabelSOM, and uses cluster centroids to find the most relevant keywords related to the cluster.

An information theoretic based approach for phrase extraction from multi-documents has been proposed by Bakus *et al* [6]. The method finds hierarchical combinations of statistically significant phrases. The approach was mainly used for document classification, but was also used to support document clustering [5].

Another method based on hierarchical clustering was proposed by Osdin *et al* [89]. Summarization of clusters is performed by extracting the phrases that best match a user-query.

Mani and Bloedorn suggested a method for multi-document summarization based on a graph representation based on concepts in the text [79]. Also another system for topic identification is TopCat [17]. It uses a series of natural language processing, frequent itemset analysis, and clustering steps to identify the topics in a document collection.

2.1.4 Document Clustering Evaluation Criteria

To evaluate the effectiveness of some method, we need accurate evaluation measures. Some of these evaluation measures are presented here, and are put specifically into the context of document clustering, as this is a major focus of the proposed work, and will be used to evaluate some of the methods used.

The results of any clustering algorithm should be evaluated using an informative quality measure that reflects the "goodness" of the resulting clusters. The evaluation depends on whether we have prior knowledge about the classification of data objects; *i.e.* whether we have labeled data, or no classification for the data is known. If the data is not previously classified we have to use an *internal quality* measure that allows us to estimate the validity of clusters without reference to external knowledge. On the other hand, if the data is labeled, we make use of this classification by comparing the resulting clusters with the original classification (ground truth); such measure is known as an *external quality* measure. We review two external quality measures, F-measure and Entropy, and two internal quality measure, Dunn's Index and Separation Index.

F-measure

This measure combines the *Precision* and *Recall* ideas from the Information Retrieval literature. The precision and recall of a cluster c_j with respect to a class (topic) t_i are defined as:

$$P = \text{Precision}(t_i, c_j) = \frac{N_{t_i c_j}}{N_{c_j}} \tag{2.9a}$$

$$R = \text{Recall}(t_i, c_j) = \frac{N_{t_i c_j}}{N_{t_i}} \tag{2.9b}$$

where $N_{t_i c_j}$: is the number of members of class t_i in cluster c_j,

N_{c_j} : is the number of members of cluster c_j, and

N_{t_i} : is the number of members of class t_i.

The F-measure of class t_i is defined as:

$$F(t_i) = \frac{2PR}{P+R} \tag{2.10}$$

With respect to class t_i we consider the cluster with the highest F-measure to be the cluster c_j that maps to class t_i, and that F-measure becomes the score for class t_i. The overall F-measure for the clustering result $\mathbf{C}$ is the weighted average of the F-measure for each class t_i:

$$F_{\mathbf{C}} = \frac{\sum_i [N_{t_i} \times F(t_i)]}{\sum_i N_{t_i}} \tag{2.11}$$

where N_{t_i} is the number of objects in class t_i. The higher the overall F-measure, the better the clustering, due to the higher accuracy of the clusters mapping to the original classes.

Entropy

Entropy indicates how homogeneous a cluster is. Lower entropy indicates more homogeneous clustering.

For every cluster c_j in the clustering result $\mathbf{C}$ we compute p_{ij}, the probability that a member of cluster c_j belongs to class t_i. The entropy of each cluster c_j is calculated using the standard formula $E_{c_j} = -\sum_i p_{ij} \log(p_{ij})$, where the sum is taken over all classes. The total entropy for a set of clusters is calculated as the sum of entropies for each cluster weighted by the size of each cluster:

$$E_{\mathbf{C}} = \sum_{j=1}^{N_C} (\frac{N_{c_j}}{N_D} \times E_j) \tag{2.12}$$

where N_{c_j} is the size of cluster c_j, and N_D is the total number of data objects.

Dunn's Index

Dunn's Index is a cluster validity index that accounts for cluster diameters and inter-cluster distances [31]. It is defined as:

$$DI = \min_{1 \leq i \leq N_C} \left\{ \min_{i < j \leq N_C} \left\{ \frac{\text{dist}(c_i, c_j)}{\max_{1 \leq i \leq N_C}(\text{diam}(c_k))} \right\} \right\} \tag{2.13}$$

where

$$\begin{aligned} \text{dist}(c_i, c_j) &= \min_{x \in c_i, y \in c_j} \{\text{dist}(x, y)\} \\ \text{diam}(c_k) &= \max_{x, y \in c_k} \{\text{dist}(x, y)\} \end{aligned}$$

Dunn's index rewards well-separated clusters that have small diameters, and thus larger values indicate better clustering solutions. A widely criticized drawback of this index is its expensive computation, especially for large N_C and N_D. Moreover, it is sensitive to noisy data points, since inter-cluster distance is based on the minimum pair-wise distance between points in different clusters.

Separation Index

Separation Index is another cluster validity measure that utilizes cluster centroids to measure the distance between clusters, as well as between points in a cluster to their respective cluster centroid. It is defined as the ratio of average within-cluster variance (cluster scatter)

to the square of the minimum pair-wise distance between clusters:

$$\begin{aligned} SI &= \frac{\sum_{i=1}^{N_C} \sum_{x_j \in c_i} \text{dist}(x_j, m_i)^2}{N_D \min_{\substack{1 \le r,s \le N_C \\ r \neq s}} \{\text{dist}(m_r, m_s)\}^2} \\ &= \frac{\sum_{i=1}^{N_C} \sum_{x_j \in c_i} \text{dist}(x_j, m_i)^2}{N_D \cdot \text{dist}_{\min}^2} \end{aligned} \tag{2.14}$$

where m_i is the centroid of cluster c_i, and $\text{dist}_{\min}$ is the minimum pair-wise distance between cluster centroids. Clustering solutions with more compact clusters and larger separation have lower Separation Index, thus lower values indicate better solutions. This index is more computationally efficient than Dunn's index, and is less sensitive to noisy data.

2.2 Clustering Algorithms

In this section some of the methods that have been reported in the literature on data clustering are presented. By definition clustering is an unsupervised learning technique, and that will be the focus of this section. However, references to some work on supervised and semi-supervised learning approaches will be given as needed.

Jain *et al* [57, 58] cover the topic very well from the point of view of cluster analysis theory. They break down the methodologies mainly into *partitional* and *hierarchical* clustering methods. Chakrabarti [15], on the other hand, categorizes clustering methods into partitioning, geometric embedding, and probabilistic approaches. We base our discussion here on Chakrabarti's categorization.

The motivation behind unsupervised learning approaches, especially clustering, is that it can often provide an automatic organization for a document collection. Topic directories, such as Yahoo! and the Open Directory (`dmoz.org`) are of great value due to their

organization of knowledge into a hierarchical taxonomy. These directories are created manually today. Clustering can help in the organization of such directories by performing a preliminary clustering, which may be followed by classification of new documents into the pre-clustered topics. Other applications include the clustering of query search results, thus providing the user with groups of results instead of a linear ranked list, which should be easier for the user to "zoom in" on their topic of interest.

2.2.1 Partitioning Clustering Approaches

One approach to clustering is to partition the document collection into k clusters $C_1, \ldots, C_k$ such that the intra-cluster distance $\sum_i \sum_{d_1,d_2 \in D_i} \mathrm{dis}(d_1, d_2)$ (where dis is the distance function) is minimized, or the intra-cluster similarity $\sum_i \sum_{d_1,d_2 \in D_i} \mathrm{sim}(d_1, d_2)$ (where sim is the similarity function) is maximized. This kind of clustering relies on external representation using pair-wise document similarity information only. If an internal representation of documents is available, such as the vector space model, then it is possible to represent a prototype of the cluster itself using, for example, the centroid of the cluster. In such cases, the goal could be to partition the collection into $C_1, \ldots, C_k$ so as to minimize $\sum_i \sum_{d \in D_i} \mathrm{dis}(d, \overrightarrow{D}_i)$ or maximize $\sum_i \sum_{d \in D_i} \mathrm{sim}(d, \overrightarrow{D}_i)$, where $\overrightarrow{D}_i$ is the vector-space representation of cluster C_i.

k-means and FCM

Partitioning could also be achieved by having a map of document-to-cluster assignment, in which assigning a document d to cluster i is done by setting a boolean variable $z_{d,i}$ to 1. In this case, we need to find a map that will minimize $\sum_i \sum_{d \in D} z_{d,i} \mathrm{dis}(d, \overrightarrow{D}_i)$ or maximize $\sum_i \sum_{d \in D} z_{d,i} \mathrm{sim}(d, \overrightarrow{D}_i)$. This is realized through the popular k-means clustering algorithm. Complexity this type of algorithms is $O(kndT)$, where k is the number of clusters, n is

the number of documents, d is the dimension of the feature space, and T is the number of iterations.

A variant of k-means that allows overlapping clusters is known as *Fuzzy C-means* (FCM). Instead of having binary membership of objects to their respective clusters, FCM allows for varying degrees of object memberships [59]. Krishnapuram *et al* [70] proposed a modified version of FCM called "Fuzzy C-Medoids" (FCMdd) where the means are replaced with medoids. They claim that their algorithm converges very quickly and has a worst case of $O(n^2)$ and is an order of magnitude faster than FCM.

Top-down vs. Bottom-up

Partitions could be created *bottom-up*, in which each document is assigned to its own cluster, and then clusters are combined in a greedy manner according to the most similar pair of clusters at each iteration, thus forming a hierarchy of clusters. This type of clustering is best known as *Hierarchical Agglomerative Clustering* (HAC). Time complexity of such algorithms is $O(n^2)$. The other approach is to set the number of clusters a priori, and create a random partition of clusters, then refine the clusters so as to satisfy one of the cost function mentioned above. Such techniques are known as *top-down* approaches.

Jain and Dubes [57] give a comprehensive account of clustering techniques including partitioning clustering.

2.2.2 Geometric Embedding Approaches

Some clustering approaches work by projecting the problem space into a two or three-dimensional space, so as to aid the user in spotting natural clusters. The merit of such approaches is that they allow the visualization of clusters, which is often considered an advantage.

Self-Organized Maps

One such approach is *Self-Organizing Maps* (SOM) [55, 56, 65, 73]. In SOM, clusters are laid out on a plane in a regular grid, and documents are iteratively assigned to regions of the plane. Clusters are *embedded* in a low-dimensional space, where the process tries to place related clusters close together in that space. Like k-means SOM associates a representative vector μ_c with each cluster c, and iteratively refines these representative vectors. Unlike k-means, each cluster is represented in the low-dimensional space.

Multidimensional Scaling (MDS)

In MDS the system input is the pairwise (dis)similarity between documents, rather than the internal vector-space representation of the documents. The algorithm seeks to project the documents onto a low-dimension space (often 2D or 3D) with minimum distortion of the original pairwise distances. This is usually keeping the Euclidean distance between any pair of points in the low-dimensional space as close as possible to the distance between them specified by the input.

Let $d_{i,j}$ be a (symmetric) user-defined measure of distance (or similarity) between documents i and j, and $\hat{d}_{i,j}$ be the Euclidean distance between the point representation of the two documents chosen by the MDS algorithm. The *stress* of the embedding (which we would like to minimize) is given by

$$\text{stress} = \frac{\sum_{i,j}(\hat{d}_{i,j} - d_{i,j})^2}{\sum_{i,j} d_{i,j}^2}$$

Convergence of this function is often difficult to achieve, and is usually done using iterative relaxation (hill climbing). Initially points are assigned random coordinates, and are moved iteratively by small distance in a direction that locally minimizes the stress.

Latent Semantic Indexing (LSI)

LSI uses techniques from linear algebra to factor the term-document matrix. The factors can be used to derive a low-dimensional representation for documents as well as terms. LSI maps synonyms and related words to similar vectors, thus bridging the syntax gap. The technique works by decomposing the term-document matrix to compute its *singular value decomposition* (SVD). The resultant singular values are ranked such that the top r singular values capture the "signal" in the original matrix, leaving out the lower singular values to account for the "noise". Berry *et al* [9] give a detailed account of LSI.

2.2.3 Probabilistic Approaches

One of the issues of the approaches discussed above is that they are considered somewhat sensitive to the similarity measure used for clustering. The probabilistic approach assumes that the documents follow a specific distribution, which the approach tries to model by finding the distribution parameters. Effectively, estimating these parameters *is* the clustering process itself.

Such approaches include the *Maximization Expectation* algorithm (based on a Gaussian Mixture Model), *Probabilistic LSI*, and Multiple Cause Mixture Model (MCMM). Some are discussed in Mitchell's standard text [84]. One of the major drawbacks of probabilistic approaches is that they are computationally expensive.

Statistical methods have also been widely used in problems related to document classification. They mostly rely on Bayesian statistics, of which the most widely used approach is the naive Bayes classifier. Primary applications include key-phrase extraction from text documents [37], text classification [18], text categorization [30], and hierarchical clustering [53, 63].

2.3 Distributed Data Mining

Distributed Data Mining (DDM) started to gain attention during the late nineties. Although it is still a young area of research, the body of literature on DDM constitutes a sizeable portion of the broader data mining literature. DDM in general deals with the problem of finding patterns in an environment where data is either naturally distributed, or could be artificially partitioned across computing nodes. It implies distribution of one or more of: users, data, hardware, or mining software [71]. Centralized data mining systems do not address some the requirements of distributed environments, such as scalability and cooperation.

Data mining in distributed environments is known as Distributed Data Mining (DDM), and sometimes as Distributed Knowledge Discovery (DKD). The central assumption in DDM is that data is distributed over a number of sites, and that it is desirable to derive, through data mining techniques, a global model that reflects the characteristics of the whole data set.

Applications of DDM are numerous, and are usually manifested as distributed computing projects. They often try to solve problems in mathematics and science. Specific areas and example projects include: astronomy (SETI@home), biology (Folding@home, Predictor@home), climate change (CPDN), physics (LHC@home), cryptography (distributed.net), and biomedicine (grid.org). Those projects are usually built on top of a common platform providing low level services for distributed or grid computing. Examples of those platforms include: Berkeley Open Infrastructure for Network Computing (BOINC), Grid.org, World Community Grid, and Data Mining Grid.

A number of challenges (often conflicting) arise when developing DDM methods:

- Communication model and complexity

- Quality of global model
- Privacy of local data

It is desirable to develop methods that have low communication complexity, especially in mobile applications such as sensor networks, where communication consumes battery power. Quality of the global model derived from the data should be either equal or comparable to a model derived using a centralized method. Finally, in some situations when local data is sensitive and not easily shared, it is desirable to achieve a certain level of privacy of local data while deriving the global model.

Although not yet proven, usually deriving high quality models requires sharing as much data as possible, thus incurring higher communication cost and sacrificing privacy at the same time.

2.3.1 Homogeneous vs. Heterogeneous distributed data

We can differentiate between two types of data distribution. The first is *homogeneous*, where data is partitioned horizontally among the sites; *i.e.* each site holds a subset of the data. The second is *heterogeneous*, where data is partitioned vertically; *i.e.* each site holds a subset of the attribute space, and the data is linked among sites via a common key.

2.3.2 Exact vs. Approximate DDM algorithms

A DDM algorithm can be described as either *exact* or *approximate*. Exact algorithms produce a final model identical to a hypothetical model generated by a centralized process having access to the full dataset. Figure 2.2 illustrates the hypothetical process that is modeled by an exact distributed clustering algorithm. The exact algorithm works as if the data subsets, D_i, from each node were brought together into one data set, D, first;

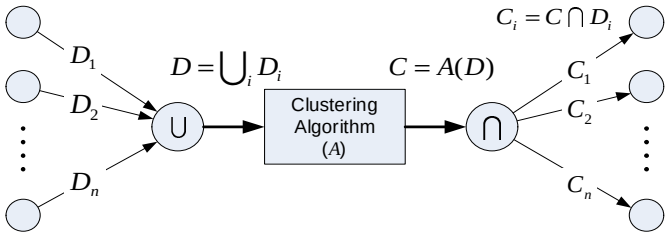


Figure 2.2: Exact Distributed Clustering Model

then a centralized clustering algorithm, $\mathcal{A}$, had performed the clustering procedure on the whole data set. The clustering solutions are then distributed again by intersecting the data subsets with the global clustering solution.

Approximate algorithms, on the other hand, produce a model that closely approximates a centrally-generated model. Most DDM research focuses on approximate algorithms as they tend to produce comparable results to exact algorithms with far less complexity [22].

2.3.3 Communication models

Communication between nodes in distributed clustering algorithms can be categorized into three classes (in increasing order of communication cost)

- **communicating models**, which involves calculating local models that are then sent to peers or a central site. These models often are comprised of cluster centroids, *e.g.* P2P k-means [25], cluster dendograms, *e.g.* RACHET [93], or generative models, *e.g.* DMBC [83];

- **communicating representatives**, in which nodes select a number of representative samples of the local data to be sent to a central site for global model generation, such as the case in the KDEC distributed clustering algorithm [67], and the DBDC

algorithm [60]; and

- **communicating data**, in which nodes exchange actual data objects; *i.e.* data objects can change sites to facilitate construction of clusters that exist in certain sites only, such as the case in the collaborative clustering scheme in [45], and the distributed signature-based clustering in [76].

2.3.4 Distributed Text and Web Mining

Applications of DDM in the text mining area are rare, but usually employ a form of distributed information retrieval. Distributed text classification and clustering have received little attention. PADMA is an early example of parallel text classification [63].

The work presented by Eisenhardt *et al* [32] achieves document clustering using a distributed peer-to-peer network. They use the k-means clustering algorithm, modified to work in a distributed P2P fashion using a probe-and-echo mechanism. They report improvement in speed up compared to centralized clustering. Their algorithm is an exact algorithm, although it requires global synchronization at each iteration.

A similar system can be found in [76], but the problem is posed from the information retrieval point of view. In this work, a subset of the document collection is centrally partitioned into clusters, for which "cluster signatures" are created. Each cluster is then assigned to a node, and later documents are classified to their respective clusters by comparing their signature with all cluster signatures. Queries are handled in the same way, where they are directed from a root node to the node handling the cluster most similar to the query.

Centralized mining can hardly scale to the magnitude of the data on the Web. Google, for example, is able to index the Web daily and respond to millions of queries per day

because it employs a farm of distributed computing nodes that apply distributed algorithms for content indexing and query processing.

2.3.5 State of the art

In the latest issue of IEEE Internet Computing [72] (at the time of writing this thesis), a few algorithms were presented representing the state of the art in DDM. Datta *et al* [23] described an exact local algorithm for monitoring a k-means clustering (originally proposed by Wolff *et al*in [106]), as well as an approximate local k-means clustering algorithm for P2P networks (originally proposed by Datta *et al*in [24, 25]).

Although the k-means monitoring algorithm does not produce a distributed clustering, it helps a centralized k-means process know when to recompute the clusters by monitoring the distribution of centroids across peers, and triggering a re-clustering if the data distribution significantly changes over time.

On the other hand, the P2P k-means algorithm in [24, 25] works by updating the centroids at each peer based on information received from their immediate neighbors. The algorithm terminates when the information received does not result in significant update to the centroids of all peers. The P2P k-means algorithm finds its roots in a parallel implementation of k-means proposed by Dhillon and Modha [27].

To summarize the literature on distributed clustering, figure 2.3 illustrates the various distributed clustering algorithms and their roots in a time line format. Entities with thick borders are distributed algorithms, while plain entities are centralized algorithms or theories behind the distributed algorithms.

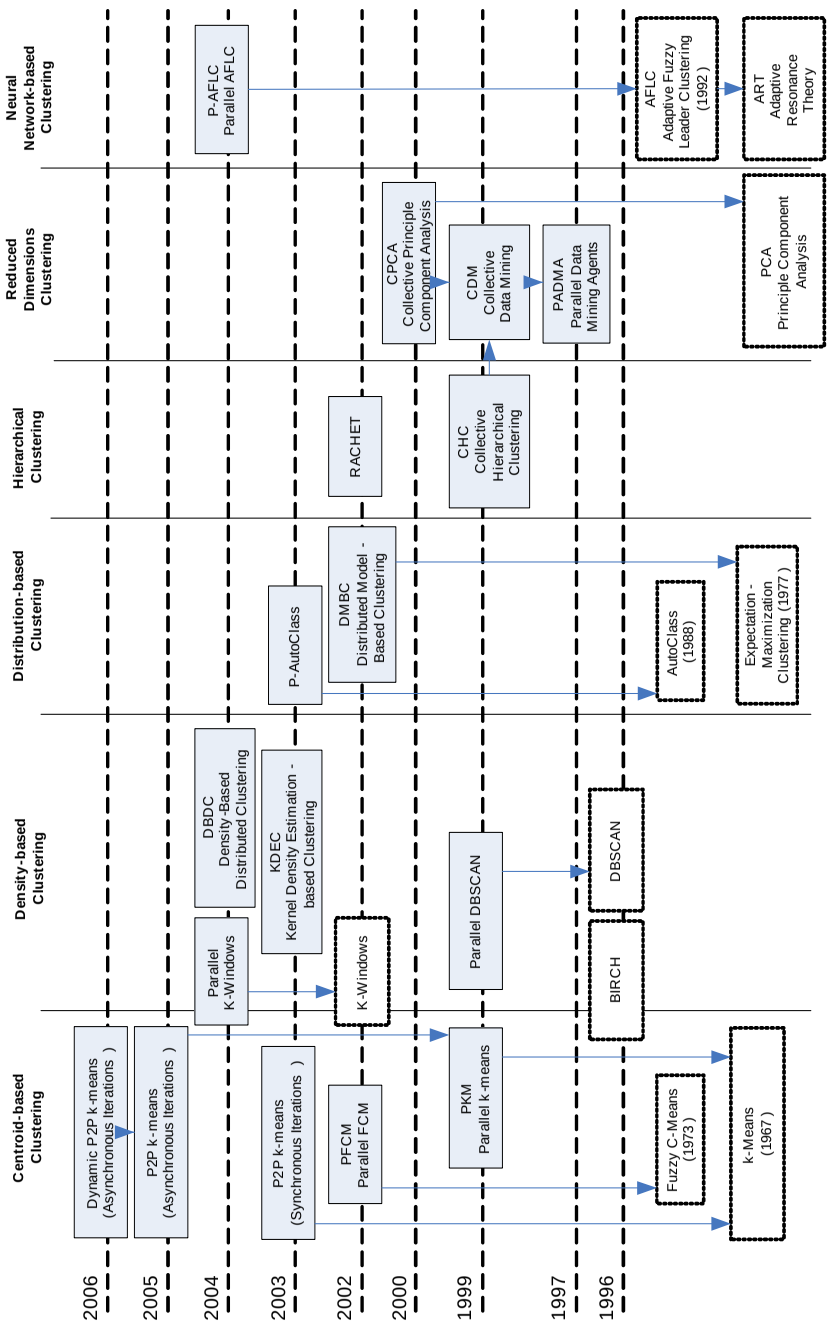


Figure 2.3: Distributed Clustering Algorithms

2.4 Agent-Based Data Mining

Data mining is often viewed from, or implemented within, an agent paradigm. A data mining process could be often viewed as an agent. The agents literature is so diverse and sometimes confusing on what is actually an agent. There has been so many theories formed to describe the underlying models driving agent behavior and reasoning. In their key 1995 article, Wooldridge and Jennings [108] give a very detailed and thorough review of the different agent theories, architectures, and languages. D'Inverno and Luck [28] also presented detailed formal agent specification framework called SMART, which describes agency in a thorough formal framework.

For simplicity it will be assumed that an agent is a software entity that possesses the ability to perform a task given certain resources. Abilities such as communication and negotiation are fundamental for a multi-agent system that will exhibit some kind of collaboration for the ultimate success of the whole system. This particular aspect of agenthood is one of the fundamental issues addressed in this research. The key is to have collaboration between a set of Web mining agents, so as to achieve maximum overall system performance.

Green *et al* [38] identify three types of agents: user interface agents, distributed agents, and mobile agents. The type associated with the topic at hand is the distributed agents type. Distributed agents technology is concerned with problem solving by a group of agents, and is most relevant here for the problem of collaborative knowledge discovery. Kargupta *et al* [63] specifically addressed this problem using a distributed architecture for data mining in general. However, the literature is quite sparse in this specific area. There has been little work on real distributed content mining using a multi-agent approach specifically to address the collaboration and scalability issues.

The most relevant approaches either use agents to aid the users in information retrieval based on their personal preferences, or use collaborative techniques to make use of other users' judgements in rating content for making recommendations.

Delgado [26] classifies agents by the underlying information filtering technology into: content-based filters, reputation-based filters, collaborative or social-based filters, event-based filters, and hybrid filters. Each one of these filtering methods map to one of either content-based mining, structure-based mining, and usage-based mining.

2.4.1 Intelligent Mining Agents

Generally speaking, agent-based data mining systems (especially for text/web mining) can be categorized as follows [19]:

Intelligent Search Agents. Several approaches are reported that search for relevant information using domain characteristics and user profiles to organize and interpret the discovered information. They mainly rely on either pre-specified domain information about particular types of documents, or hard coded models of the information sources to retrieve and interpret the documents.

Information Filtering/Categorization. Some agent-based approaches use various information retrieval techniques and characteristics of hypertext documents to automatically retrieve, filter, and categorize those documents. Hypersuit [103] is one such system. It uses semantic information embedded in link structures and document content to create cluster hierarchies of hypertext documents, and structure an information space.

Personalized Web Agents. These agents learn user preferences and discover Web information sources based on these preferences, and those of other individuals with similar interests (using collaborative filtering).

2.4.2 Multi-Agent Systems

D'Inverno and Luck [28] define a multi-agent system as one that contains a collection of two or more agents, with at least one autonomous agent, and at least one relationship between two agents where one satisfies the goal of the other. The requirement for an autonomous agent in the system was based on the argument that the system should have some kind of goal generation mechanism, which is provided by autonomous agents only through motivation.

Multi-agent systems usually operate in distributed and complex environments. In a typical collaborative multi-agent system, each agent has incomplete information or capabilities. These agents work together to achieve a global objective based on distributed data, mostly in a decentralized fashion.

To enable effective inter-agent communication and coordination, agents that work together have to use an inter-operable, platform-independent, and semantically unambiguous communication protocol. There are two widely used such protocols, called Agent Communication Language (ACL): the Knowledge Query and Manipulation Language (KQML) and the FIPA ACL. KQML, developed as part of the ARPA Knowledge Sharing Effort, is a language and protocol for exchanging information and knowledge among software agents. In KQML, each expression is a speech act described by a performative. The FIPA ACL was developed by the Foundation for Intelligent Physical Agents (FIPA). Similarly to KQML, the FIPA ACL is based on speech act theory and the two languages have similar syntax.

Another paradigm for agent communication which has been gaining momentum recently is Web Services, which is based on the standard XML data format for exchanging information between agent-like programs on the Web. A Web service is an entity that provides reusable functionality accessible over the Web through the standard SOAP[4] protocol.

[4]Simple Object Access Protocol

This platform allows for the discovery and integration of distributed services over the Web, so as to build a complete systems from reusable components. A multi-agent system using Web services for communication will be able to seek and reuse other agents capabilities to satisfy its goals.

In general, the agent paradigm lends itself to distribution very well. Attributes such as autonomy, cooperation, negotiation, and sharing are of great value to a distributed mining system.

2.5 Application Scenario: E-learning

Educational Data Mining is an emerging discipline, mainly focusing on the types of data consumed or produced by learners in e-learning environments. This field of study is commonly known as Artificial Intelligence in Education (AIED), and most of the work in the literature in this area uses various AI techniques in the context of e-learning to design what is known as Intelligent Tutoring Systems (ITS). The application of data mining methods to e-learning is assumed to help in understanding and improving the student and/or teacher experience through discovering previously unknown trends or patterns [46].

According to Monk [85], the best use of online media for e-learning, such as the web, is not simple delivery of printed material, but rather for collaboration and discussion, simulation and testing, tutoring and guidance, and feedback. He also argues that learning environments should be structured around learning outcome, not content. Under these insightful remarks, we can see that data mining, as a tool for discovery, extraction, and adaptation, can be of great benefit to e-learning.

2.5.1 Actors and Goals

We can view e-learning environments as multi-actor systems, where agent-like actors are part of distributed activities. We can identify three types of such actors who can benefit from data mining in e-learning: learners, instructors, and learning designers. Although they have correlated roles, their goals could be substantially different and on varying scale.

Learner

Learners (students) are the main focus of many e-learning studies. Data mining methods are mainly used to *aid* the learner. Notable methods that help in guiding the learner are classification methods for predicting learner outcome, failure risk analysis, self assessment, and providing suggestions for improving competencies. In those scenarios, data mining is typically applied to learner data that spans weeks, to the extent of a full semester.

Instructor

Instructors are the primary direct user of data mining tools in e-learning. Although learners may benefit indirectly through data mining, it is the instructor who proactively utilizes data mining tools for the purpose of identifying patterns among learners. Among other goals, instructors may use data mining to evaluate either the learners or the learning process itself. They are able to find patterns among groups of learners (through clustering techniques), or identify a bottleneck in the course instruction process (through pattern mining of course activity events or student discussions). Typically, the data used in those scenarios spans months, to the extent of a few semesters in which the course is offered.

Learning Designer

Learning designers are involved in designing courses or instructional processes. Data mining can help learning designers in two aspects: understanding learner behavior, and gaining insight into good course structure and composition. Data in such scenarios come from learner modeling systems, which typically spans years.

2.5.2 Types of Data

Before delving into data mining scenarios, it is imperative to describe the types of data usually subject to analysis in e-learning environments. The most visible types of data in educational data mining are learner interaction logs, and course content and its associated metadata. We discuss each of those types briefly in this section.

Interaction Logs

The most common data type subject to analysis and data mining in e-learning is learner interaction logs. Usage logs usually describe the nature of interaction, such as the type of activity, duration of activity, transition between activities, and various context information.

Muehlenbrock [86] describes that this type of data is similar to web site access logs, but argues that in web access scenarios the available logs are shallow, providing only click-through streams of information, while in learning settings more pedagogical data is available for logging and thus should be leveraged during analysis.

Interaction logs are usually used for the purpose of user modeling. By identifying regularities in usage logs, a descriptive model of the learners is derived that can be used to predict future learner actions. Brooks *et al* [14] describe a Massive User Modeling System (MUMS) to support the modeling of learners in distributed e-learning environments. The

system utilizes a broker model for event capturing and distribution. Learner interactions are captured by MUMS-compliant modules, such as message boards and learning management systems, and sent to the central MUMS server where they are made available to subscribers for analysis. They do not imply how the interaction data should be analyzed to model learner behavior; their aim was to effectively collect and store pedagogical events for later consumption by modelers and reasoners in a scalable distributed learning environment.

Content and Metadata

E-learning systems follow strict standards for static course content structure and organization, as well as dynamic sequencing of course elements during online instruction. The *IMS Content Packaging* (IMS CP) standard is used for specifying how course modules, lessons, and learning objects are put together in a hierarchical structure suitable for course organization.

Usually associated with the course structure are *metadata* that describe the content, such as the title, author, publisher, subject, and copyright information. Metadata standards in use by e-learning systems include Dublin Core and IEEE Learning Object Metadata (IEEE LOM). The role of metadata is to provide cataloging information that can be used to make search and access to the content easier and more efficient.

Metadata can also describe pedagogical information, such as competency levels, target learner class, duration of instruction; it also can describe technical aspects, such as format of the material and the system requirements to access it.

The literature on content mining for e-learning is quite scarce. Since tagging content with metadata is an expensive process, most of the work in this area is focused on automatically generating metadata from content and other context data. Brooks *et al* [13] discussed

the issues of automatic metadata generation through collaborative filtering techniques, as well as by directly applying text mining to the content for the purpose of metadata extraction. The system can accurately extract useful metadata from content, although it is limited to a few metadata fields.

2.5.3 Data Mining Scenarios

We can broadly divide the work in educational data mining into two categories: usage pattern mining, and extraction of information.

Pattern Mining

The interaction between learners and e-learning systems produce a large amount of data, which is usually captured into logs for further analysis. It is the job of usage pattern mining to analyze such logs. The goal of this process is manifold, but usually is focused on detecting commonalities or anomalies in usage. Commonalities are similarity in usage patterns that can help instructors identify group behavior.

Merceron and Yacef specifically designed a tool, called TADA-Ed, for mining student interaction with an online tutoring system [80, 81]. The tool incorporates two clustering algorithms, k-means and hierarchical clustering and one decision tree classifier, all adapted from open source data mining library Weka [105]. It also implements the popular association rule algorithm *a priori*, with an adapted version that takes sequences of events into consideration. Their primary goal was to identify frequent mistakes done by students, and consequently identify either a group of students, through clustering, who need further attention, or identify certain topics in the curriculum that may need modification. The classifier was used to predict student marks given current context and previous history. Although the algorithms they implemented are basic data mining algorithms, it clearly

shows the benefit of data mining when correctly applied to an application domain such as e-learning.

A similar attempt has been made by Superby *et al* [99], where they analyzed performance data collected about first year university students with the purpose to predict those at risk of failure, and thus enabling them to take corrective measures so as to avoid this risk. They mainly used classification techniques such as discriminant analysis, neural networks, and decision trees. Their data primarily come from a questionnaire administered to first year students, which is used in correlation with their academic performance early in the year.

Other work in pattern mining for e-learning was done by Kay *et al* [66]. Their work was fundamentally aimed at assessing teamwork processes and providing feedback to students about how they could improve them. Their data were collected through a system called TRAC for tracking different types of events in an e-learning environment. They used a frequent sequential pattern mining algorithm, which is based on the *a priori* pattern mining algorithm. Although they were able to mine many patterns from the data, they were not able to interpret them properly, and conclude that further clustering of patterns may be required to properly differentiate classes of patterns.

Federated Search

Educational content is usually stored and indexed by digital repositories, called Learning Object Repositories (LORs), which facilitate storage and retrieval. The existence of multiple distributed repositories creates a problem if we would like to look for content that could be available in any of those repositories. A possible solution is the *federated search* concept, in which repositories are peered together, and search across repositories is performed in a peer-to-peer manner.

Hatala *et al* [50] developed an interoperability protocol, the EduSource Communication Layer (ECL), that links digital repositories. ECL implements the IMS Digital Repository Interoperability Standard (IMS DRI). Through ECL, different repositories, with different metadata schemas, can peer together, and answer requests in a peer-to-peer manner for storage and retrieval of learning objects.

Collaborative Filtering

Collaborative filtering is a way to tap into the collective wisdom of the crowd. By analyzing the preferences or traces of a large enough number of individuals, we can filter the most common preferences and suppress the unpopular ones. For example, Brooks *et al* [13] suggested a method by which metadata tags assigned by students to content can be filtered by keeping the most frequent tags. This way, they did not need expensive manual labeling by experts, although the quality of the filtered tags could be questioned. The same method could be used to filter content based on recommendation by a large number of students, which provides an automatic way to promote high quality content simply by allowing students to vote for or against certain learning materials.

Winters *et al* [101] also suggested that collaborative filtering could be used to analyze test scores by constructing a matrix recording scores for every student and every question. Through collaborative filtering techniques, we can discover what the fundamental topics of a course are and the proficiencies of each student in those topics.

2.6 Summary

The various fields relevant to the research in this thesis have been reviewed so as to give the reader the necessary background to follow the work introduced in later chapters. Specif-

ically, background on text mining, clustering algorithms, distributed data mining, and agent-based data mining has been discussed. One specific application area of data mining (e-learning) has been reviewed briefly so as to link basic research to real-world application scenarios.

The background on text mining will be of benefit in chapter 3, where the keyphrase extraction algorithm is introduced. The background on distributed data mining and agent-based data mining will be of benefit when discussing chapters 4 and 5, where the collaborative peer-to-peer clustering and hierarchically-distributed clustering methods are discussed. The background on clustering in general will be of benefit throughout the thesis, since it is the backbone of the work introduced herein.

CHAPTER 3

Document Cluster Summarization Using Keyphrase Extraction

Effectively describing the patterns extracted through data mining is a challenging problem. First we need to identify possible patterns, then we need to select those interesting ones and describe them properly. In document clustering this means we need to describe document clusters effectively if we want to reason about them. *Keyphrase extraction* provides an answer to this problem. Automatic keyphrase extraction from document clusters provides a very compact summary of the contents of the clusters, which often helps in locating information easily.

This chapter introduces an algorithm for topic discovery using keyphrase extraction from multi-document sets or clusters based on frequent and significant shared phrases between documents [44]. The keyphrases extracted by the algorithm are highly accurate and fit the cluster topic. Subjective as well as quantitative evaluation show that the algorithm outperforms keyword-based cluster-labeling algorithms, and is capable of accurately

discovering the document cluster topic.

While this algorithm on its own is useful for labeling document clusters, it is used in the cluster summarization step of the collaborative clustering algorithm described in chapter 4. Cluster keyphrase summaries are exactly what is used to succinctly inform remote nodes of the content of local document clusters, which in turn is used to judge the similarity between remote data and local clusters.

3.1 Overview

In this chapter we present a highly accurate method for extracting keyphrases from multi-document sets or clusters, with no prior knowledge about the documents; *i.e.* it is domain-independent. The algorithm is called **CorePhrase**, and is based on finding a set of core phrases that best describe a multi-document set.

The algorithm works by first *intersecting* all documents together to generate a list of candidate keyphrases for describing the topic of the documents. Intersecting every pair of documents is a very time-consuming task, especially if we would like to find not only shared words, but shared phrases as well. The algorithm employs a powerful phrase-based document indexing model [42, 39, 40] for pair-wise phrase matching among documents.

The extracted candidate keyphrases are then analyzed for frequency, span over the document set, and other features. Each phrase is assigned a score based on its features, then the list is ranked and the top phrases are output as the descriptive topic of the document cluster. Four scoring method variants are employed and their performance is analyzed. Figure 3.1 illustrates the different components of the keyphrase extraction system.

Results show that the extracted keyphrases are highly relevant to the topic of the document set, and accurately describe their topic. We used two data sets representative of

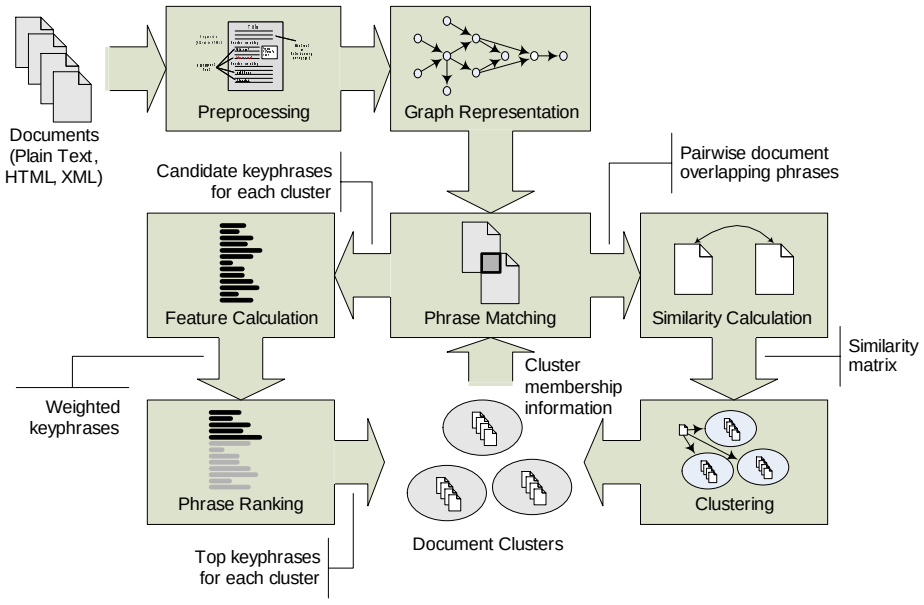


Figure 3.1: CorePhrase Keyphrase Extraction System

two keyphrase extraction tasks: one represents a collection of documents retrieved through a search engine, and the other is a collection of intranet web pages. The algorithm is so accurate that in most cases the top one or two phrases are sufficient to describe the reference topic, thus providing the end user with very compact description, rather than a long list of keywords as usually the case with keyword-based methods.

The work presented here assumes the following:

- Keyphrase extraction: keyphrases are assumed to exist in the text and are not automatically generated.
- Unsupervised extraction: the algorithm *discovers* keyphrases rather than *learns* how to extract them.
- If used with text clustering, the algorithm is not concerned with how the clusters are generated; it extracts keyphrases from clusters that have been already generated by

some clustering algorithm.

In the following section some issues in keyphrase extraction are discussed. Section 3.3 presents the CorePhrase algorithm and and discusses its complexity.

3.2 Keyphrase Extraction

Keyphrase extraction algorithms fall into two categories: keyphrase extraction from individual documents, and keyphrase extraction from a set of documents. The difference is subtle, but reflects two different problems, which sometimes overlap.

Keyphrase extraction from a single document is often posed as a document summarization task, and usually employs supervised machine learning, where an algorithm is first trained on a set of documents to *learn* the relationship between a document and its associated set of keyphrases; then the learned model is applied to unseen documents to produce a summary for them.

On the other hand, keyphrase extraction from a set of documents is often associated with document clustering to describe the produced clusters. It is regarded as an unsupervised machine learning task, where the algorithm tries to *discover* the set of keyphrases that best describe the set of documents.

3.2.1 Extraction vs. Construction

In the literature there are two ways to finding relevant keyphrases in text: either to *extract* them from existing phrases in the text, or to automatically *construct* them [37]. In theory, construction of keyphrases is regarded as a more intelligent way of summarizing text that promises to produce more comprehensible keyphrases. In practice, the automatic

construction of keyphrases is quite difficult and there is no guarantee that the new phrases will be meaningful.

In our work we assume that the representative phrases of a document cluster should be extracted from existing phrases in the cluster rather than constructed to fit the cluster.

3.2.2 Keyphrases vs. Keywords

In the context of cluster description (or characterization), we define a *keyphrase* to be "a sequence of one or more words that is highly relevant to a cluster of text documents." A *keyword*, on the other hand, is "a single word that is highly relevant to the cluster." It should be noted that an arbitrary combination of keywords does not necessarily constitute a keyphrase; neither do the constituents of a keyphrase necessarily represent individual keywords. However, we believe that keywords and keyphrases will overlap most of the time.

That said, we argue that using a small set of (one or two) succinct keyphrases to describe a cluster of text documents is a better way, subjectively, to characterize the cluster than using a longer set of keywords. Since keyword-based cluster description methods (*e.g.* [82, 88]) often have to produce many keywords to describe the cluster, we believe that using one or two highly accurate keyphrases would be favored over such methods.

3.2.3 Evaluation of Keyphrases

In practice, to quantify the quality of extracted keywords or keyphrases we need to use extrinsic measures; *i.e.* using an external reference against which to compare the extracted phrases. Section 6.2.1 discusses two evaluation measures based on the overlap of the extracted phrases with a reference topic of the document set or cluster.

3.3 The CorePhrase Algorithm

Virtually every keyphrase extraction algorithm works by first constructing a list of candidate keyphrases, scoring each candidate keyphrase according to some criteria, ranking the keyphrases by score, and finally selecting a number of the top ranking keyphrases for output. In this section, the CorePhrase multi-document keyphrase extraction algorithm is presented.

3.3.1 Extraction of Candidate Keyphrases

A candidate keyphrase that has the power to represent a set of documents in a cluster (rather than a single document) would naturally lie at the *intersection* of those documents. The CorePhrase algorithm works by first finding all possible keyphrase candidates through matching document pairs together, extracting all matching phrases between document pairs. A master list of candidate phrases for the document cluster is then constructed from the pairwise document matching lists by consolidating the individual lists to remove duplicates. The resulting list contains all phrases that are shared by at least two documents.

This process of matching every pair of documents is inherently $O(n^2)$. However, by using a proven method of document phrase indexing graph structure, known as the Document Index Graph (DIG), the algorithm can achieve this goal in near-linear time [42]. In DIG, phrase matching is done in an incremental fashion; all documents up to document d_i are represented by a graph structure, and, upon introducing a new document d_{i+1}, the new document is matched to the graph to extract matching phrases with all previous documents. The new document is then added to the graph. This process produces complete phrase-matching output between every pair of documents in near-linear time, with arbitrary length phrases.

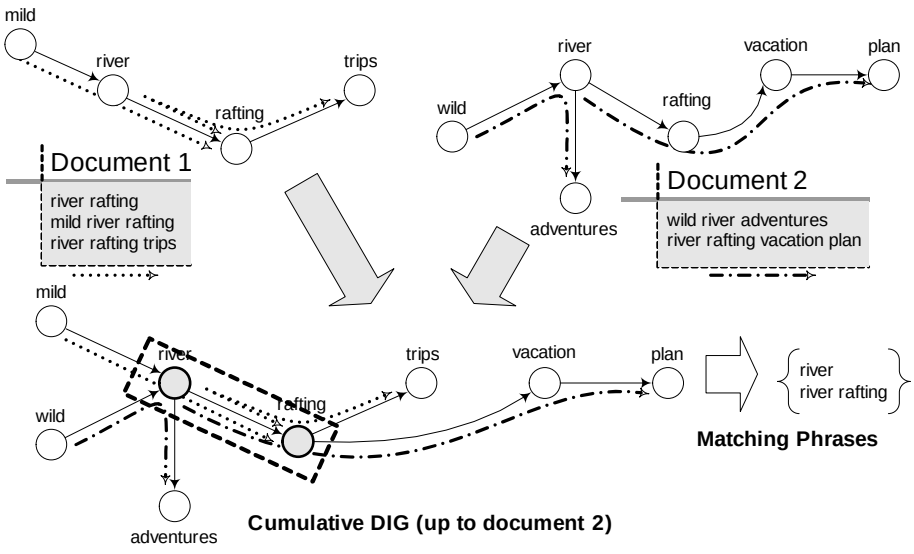


Figure 3.2: Phrase Matching Using Document Index Graph

Figure 3.2 illustrates the process of phrase matching between two documents. In the figure, the two subgraphs of two documents are matched to get the list of phrases shared between them.

The following set of definitions explain how the DIG model is used to represent documents and extract matching phrases between them.

Document Subgraph: Each document d_i is mapped to a subgraph g_i that represents this document in a stand-alone manner. Each subgraph can be viewed as a *detached* subset of the DIG that represents the corresponding document in terms of the DIG properties: $g_i = \{V_i, E_i\}$, where V_i is the set of nodes corresponding to the unique words of d_i, and E_i is the set of edges representing the sentence paths of d_i.

Cumulative DIG: Let the DIG representation of the documents processed up to document d_{i-1} be G_{i-1}, and that of the documents processed up to document

d_i be G_i. Computing G_i is done by *merging* G_{i-1} with the subgraph g_i:

$$G_i = G_{i-1} \cup g_i \tag{3.1}$$

G_i is said to be the *Cumulative DIG* of the documents processed up to document d_i.

Phrase Matching: A list of matching phrases between documents d_i and d_j is computed by *intersecting* the subgraphs of both documents, g_i and g_j, respectively. Let M_{ij} denote such list, then:

$$M_{ij} = g_i \cap g_j \tag{3.2}$$

A list of matching phrases between document d_i and all previously processed documents is computed by intersecting the document subgraph g_i with the cumulative DIG G_{i-1}. Let M_i denote such list, then:

$$M_i = g_i \cap G_{i-1} \tag{3.3}$$

When a matching phrase, l_{ij}, is found between documents d_i and d_j, we calculate its features with respect to each document, l_i and l_j, respectively, according to section 3.3.2.

Since this method outputs matching phrases for each new document, it is essential to keep a *master list*, M, of unique matched phrases, which will be used as the list of candidate keyphrases. The following simple procedure keeps this list updated:

The set of matching phrases from all documents forms a pool of candidate keyphrases. Each phrase in this pool is guaranteed to have been shared by at least two documents.

Algorithm 3.1 Extract Matching Phrases

1: {calculate M_i for document d_i using (3.3)}
   $M_{ij} = \{l_{ij}: 1 < j < i\}$: matching phrases between d_i and d_j
   $M_i = \{M_{ij}\}$: matching phrases of d_i
2: **for each** phrase l_{ij} in M_i **do**
3: **if** phrase l_{ij} is in master list M **then**
4: add feature vector l_i to l_{ij} in M
5: add feature vector l_j to l_{ij} in M if not present
6: **else**
7: add l_{ij} to M
8: add feature vectors l_i and l_j to l_{ij} in M
9: **end if**
10: **end for**
11: **for each** *unique* phrase l_k in M **do**
12: calculate averages of feature vectors associated with l_k
13: **end for**

It should be noted that using the matching phrases from multi-document sets as candidate keyphrases saves us from problems often faced by single-document keyphrase extraction, namely that of having to identify possible candidates using heuristic techniques, such as the case in the Kea [37] and Extractor [102] algorithms.

3.3.2 Phrase Features

In order to judge the quality of the candidate keyphrases, we need to differentiate between them based on quantitative features. Each candidate keyphrase l is assigned the following features:

***df*: document frequency**; the number of documents in which the phrase appeared, normalized by the total number of documents.

$$df = \frac{\mid \text{documents containing } l \mid}{\mid \text{all documents} \mid}$$

w**:** average **weight**; the average weight of the phrase over all documents. The weight of a phrase in a document is calculated using structural text cues. Examples: title phrases have maximum weight, section headings are weighted less, while body text is weighted lowest.

pf**:** average **phrase frequency**; the average number of times this phrase has appeared in one document, normalized by the length of the document in words.

$$pf = \arg avg \left[\frac{\mid \text{occurrences of } l \mid}{\mid \text{words in document} \mid} \right]$$

d**:** average phrase **depth**; the location of the first occurrence of the phrase in the document.

$$d = \arg avg \left[1 - \frac{\mid \text{words before first occurrence} \mid}{\mid \text{words in document} \mid} \right]$$

Those features will be used to rank the candidate phrases. In particular, we want phrases that appear in more documents (high df), have higher weights (high w), higher frequencies (high pf), and shallow depth. It might seem counter-intuitive to look for phrases with high df to readers familiar with the tf-idf term weighting scheme. Remember that we are not scoring the phrase with respect to a particular document, but rather with respect to the whole document set. So the more common a phrase is across all documents the higher its weight should be.

The df feature can be regarded as the *support* of the keyphrase; *i.e.* from a frequent-set analysis point of view, df tells how many items (documents) support the keyphrase. Since we are extracting keyphrases that are shared by at least two documents, the minimum support is accordingly two. Although this may seem unnecessarily low support value, when the keyphrases are ranked (as described in the next section), the top ranking phrases

usually exhibit high support.

3.3.3 Phrase Ranking

In a single-document keyphrase extraction setting, the above phrase features will be used as input vectors to a machine learning algorithm for training. The model is then applied to unseen documents to extract the keyphrases. However, in our case we are looking at discovering "good" keyphrases from multi-document data sets or clusters. Thus, we will use the features to calculate a *score* for each phrase, rank the phrases by score, and select a number of the top phrases as the ones describing the topic of the cluster.

There are two phrase scoring formulas used, as well as two methods of assigning the score to the candidate phrases, for a total of four variants of the CorePhrase algorithm.

First Scoring Formula. The score of each phrase l is calculated using the following empirical formula:

$$\text{score}(l) = (w \cdot pf) \times -\log(1 - df) \tag{3.4}$$

The equation is derived from the tf×idf term weighting measure; however, we are rewarding phrases that appear in more documents (high df) rather than punishing those phrases. Notice also that the first scoring formula does not take the *depth* feature into account. We will refer to the variant of the algorithm that uses this formula as CorePhrase-1.

Second Scoring Formula. By examining the distribution of the values of each feature in a typical corpus (see Table 6.2 for details), it was found that the *weight* and *frequency* features usually have low values compared to the *depth* feature. To take this fact into account, it was necessary to "expand" the *weight* and *frequency* features by taking their square root, and to "compact" the *depth* by squaring it. This helps even out the feature

distributions and prevents one feature from dominating the score equation. The formula is given in equation 3.5.

$$\text{score}(l) = (\sqrt{w \cdot pf} \cdot d^2) \times -\log(1 - df) \tag{3.5}$$

We will refer to the variant of the algorithm that uses this formula as CorePhrase-2.

Word weight-based score assignment. A *modified* score assignment scheme based on word weights is also used:

- First, assign *initial* scores to each phrase based on phrase scoring formulas given above.
- Construct a list of unique individual words out of the candidate phrases.
- For each word: add up all the scores of the phrases in which this word appeared to create a word weight.
- For each phrase: assign the *final* phrase score by adding the individual word weights of the constituent words and average them.

We will refer to the variants of the algorithm that use this method as CorePhrase-1M and CorePhrase-2M, based on the equation that was used to assign the initial phrase scores.

3.3.4 Complexity Analysis

The analysis of this algorithm can be divided into two stages: extraction of candidate keyphrases, and scoring and ranking the top keyphrases. As mentioned in section 3.2, candidate keyphrase extraction relies on near-linear time performance of the DIG phrase matching algorithm. This is achieved through graph matching of each newly introduced

document to the cumulative graph of all previous documents. However, this graph-based algorithm's memory requirements can be demanding in some scenarios, typically those involving long documents. Assume that:

$$\begin{aligned} n &: \text{ is the number of documents in the data set,} \\ m &: \text{ is the number of unique terms in the data set,} \\ idf_{\text{avg}} &: \text{ is the average inverse document frequency (IDF), and} \\ q &: \text{ is the average number of terms per document,} \end{aligned}$$

then the space requirements of the model is:

$$\text{size}(G) = (m \cdot idf_{\text{avg}}) + q \cdot n \tag{3.6}$$

The first term in (3.6) accounts for individual words feature storage, while the second term accounts for the extra storage required for phrase indexing structures.

Phrase scoring is inherently linear with the number of candidate keyphrases, while ranking the phrases is bound by the sorting algorithm used. We found that the processing time of this stage can be considered insignificant compared to the time required for the candidate keyphrase extraction itself.

3.4 Summary

In this chapter an unsupervised algorithm for keyphrase extraction from document clusters was presented. The algorithm, CorePhrase, works by extracting a full list of candidate keyphrases from a document cluster by matching phrases using the document index graph model. Candidate keyphrases have a support of at least two documents. The algorithm then scores each candidate phrase based on associated features, ranks the candidate

keyphrases, and finally extracts the top L keyphrases.

The algorithm works because it measures the global relevance of keyphrases to the whole cluster (coverage), as well as taking into consideration the local average significance of phrases within individual documents (weight, frequency, and depth). By balancing those two criteria, it is able to identify highly relevant phrases.

Experimental results in chapter 6, section 6.2, illustrate the high accuracy of the algorithm. For example, from a list of 24,403 candidate phrases, the top 10 keyphrases were highly relevant to the cluster topic. In fact, the top ranked keyphrase was an exact match to the reference topic (manually labeled).

In the next chapter, CorePhrase is employed within the collaborative document clustering scheme as a component for cluster summarization. Sharing only summarized cluster information (in the form of keyphrase vectors) between nodes has the desired effect of minimizing traffic between nodes, as well as providing the ability of remote nodes to perform content-based similarity calculation between the remote documents and the summarized clusters.

CHAPTER 4

Collaborative Peer-to-Peer Document Clustering

For the past few decades the mainstream data clustering technologies have been fundamentally based on centralized operation; data sets were of small manageable sizes, and usually resided on one site that belonged to one organization. Today, data is of enormous sizes and is usually located on distributed sites; the primary example being the Web. This has created a need for performing clustering in distributed environments. Distributed clustering solves two problems: infeasibility of collecting data at a central site, due to either technical and/or privacy limitations, and intractability of traditional clustering algorithms on huge data sets.

In this chapter we propose a collaborative approach for clustering documents in distributed peer-to-peer environments. The collaborative approach finds locally-optimized clusters. The main objective is to allow nodes in a network to first form independent opinions of local document clusters, then collaborate with peers to enhance the local clusters.

Information exchanged between peers is minimized through the use of cluster summaries in the form of keyphrases extracted from the clusters. This summarized view of peer data enables nodes to request merging of remote data selectively to enhance local clusters. Initial clustering, as well as merging peer data with local clusters, is based on similarity histogram-based clustering (SHC) [41], that keeps a tight similarity distribution within clusters. This approach achieves significant improvement in local clustering solutions without the cost of centralized clustering, while maintaining the initial local clustering structure.

4.1 Overview

There is a strong movement towards more distributed computing such as peer-to-peer computing and grid computing. This movement stems from two problems with centralized operation in general: the infeasibility of collecting data at a central location (due to technical and/or privacy limitations), and intractability of traditional clustering algorithms when applied to huge data sets.

In other situations involving distributed data, having one global clustering solution is not required; rather, multiple local clustering solutions are desired, each of which can then be enhanced by having access to *summarized information* from remotely distributed data. This is the situation addressed by this chapter.

As a motivating scenario, consider data distributed among a set of digital libraries. Each library can form an opinion about the topic groups in its collection by applying local clustering. To further enhance this local clustering solution, each library can receive recommendations from remote libraries on what articles, which it currently does not carry, can fit into its clustering solution. Minimum information is exchanged, but wider access to distributed data that contributes to improving local clustering is achieved.

In this chapter we present an approach that enables *collaborative clustering* among distributed P2P nodes [47]. The collaborative nature of the algorithm is a central characteristic that provides the ability of one node to benefit other nodes based on their needs.

Figure 4.1 illustrates the collaborative clustering system presented in this chapter, while Figure 4.2 illustrates the inter-node communication sequence. The network is modeled as a connected graph in which every node is connected to every other node. The document collection is equally partitioned across nodes.

The system is comprised of three main components. First, a clustering algorithm, called "Similarity Histogram-based Clustering" (SHC), is introduced. The algorithm is incremental, and is based on statistical representation of the distribution of pair-wise document similarities within a cluster. By carefully monitoring this distribution during the cluster update process we are able to guide the clustering algorithm towards producing accurate clusters. The SHC algorithm is presented in section 4.3.1.

The second component involves cluster summarization through keyphrase extraction. The purpose of this process is to minimize the amount of information being exchanged in the distributed environment. Each node creates a set of cluster summaries, in the form of keyphrase vectors, which are exchanged with its peers. The keyphrase extraction algorithm, CorePhrase, has been introduced in chapter 3.

The third component is the main collaborative document clustering algorithm. It is a distributed variant of the SHC algorithm that involves a series of steps: local cluster generation and summarization, exchange of cluster summaries, similarity calculation of cluster summaries to peer documents, recommendation of peer documents, and merging of peer documents into local clusters. The collaborative clustering algorithm is presented in section 4.3.3.

In the following section the collaborative peer-to-peer clustering model is presented.

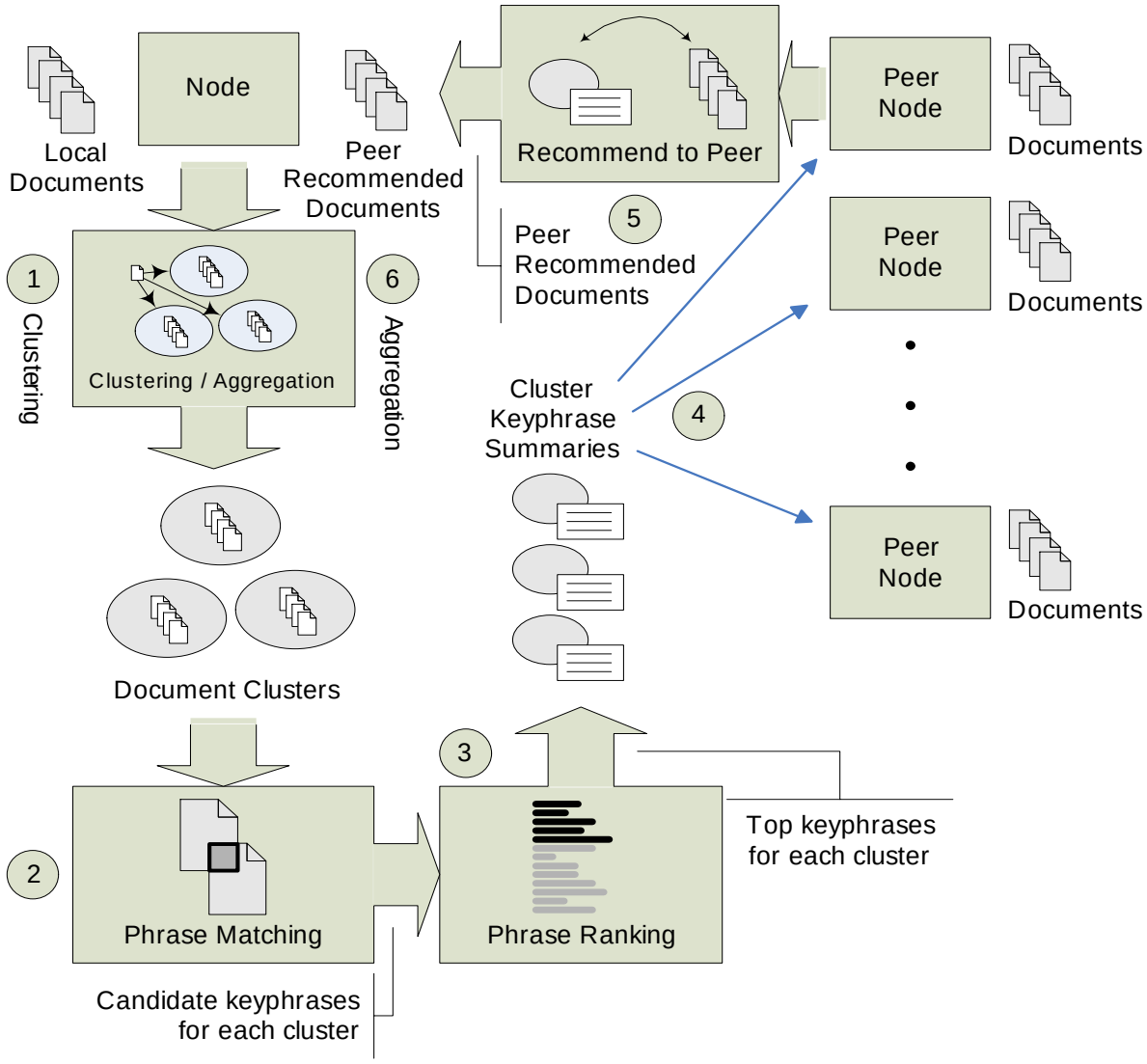


Figure 4.1: Distributed Collaborative Document Clustering System

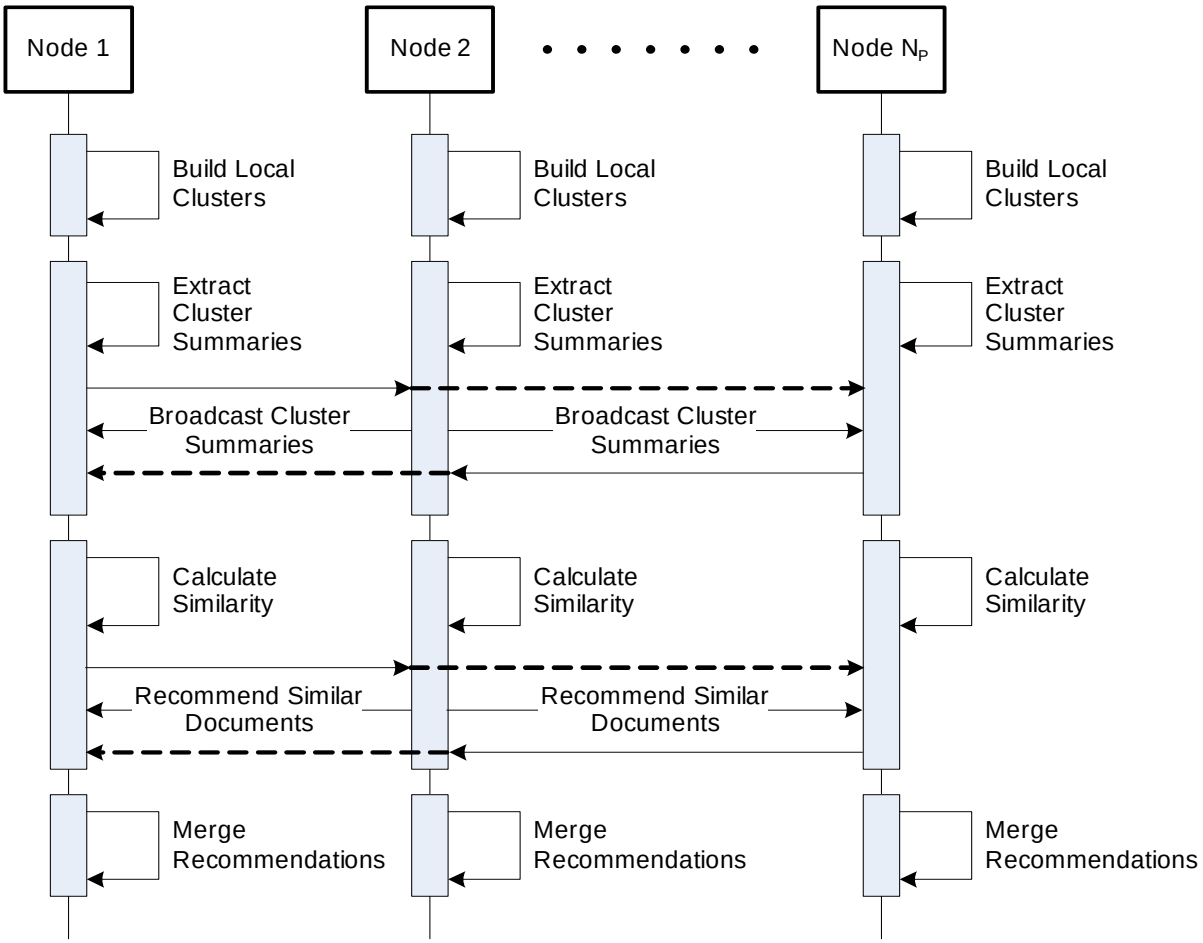


Figure 4.2: Peer-to-Peer Communication Sequence

Section 4.3 presents the algorithms for initial local clustering and subsequent distributed collaborative clustering. Finally, a summary is given in section 4.4.

4.2 Collaborative Peer-to-Peer Clustering Model

To achieve the goal of collaborative clustering, we propose a number of models that describe different aspects of the problem. In this section we discuss how to represent the distributed nodes and how the data is distributed among them, how to represent clusters of documents, and how the documents themselves are represented.

4.2.1 Distributed P2P Model

Distributed nodes form a network of peers, where each node performs exactly the same tasks; *i.e.* there is no centralized operation. The network is represented as a connected graph $\mathbf{G}(\mathbf{P}, \mathbf{L})$

where $\mathbf{P}$: is the set of *peers* or *nodes* $\{p_i\}$, $i = 1, \ldots, N_P$.

$\mathbf{L}$: is the set of *links* $\{l_{ij}\}$ between every pair of nodes p_i and p_j, $i, j = 1, \ldots, N_P$, $i \neq j$. The number of links in the network $N_L = N_P(N_P - 1)/2$.

This kind of connectedness is chosen for the sake of simplicity, since the typical scenarios for our approach involves only several nodes.

The number of nodes is static; *i.e.* N_P is fixed for a certain configuration and cannot be dynamically changed. We, however, test with different network sizes for different values of N_P.

Node links are assumed to be equally weighted. Nodes are assumed to have identical roles in the network, hence the peer-to-peer model.

Data Distribution Model

The global view of data is a collection of documents $\mathbf{D} = \{d_i\}$, $i = 1, \ldots, N_D$. The document collection is randomly, but evenly, distributed across network nodes; *i.e.* each node holds the same number of documents, which is a fixed percentage α of the total number of documents N_D. We refer to the parameter α as the *distribution ratio*. Overlap between the nodes in terms of the documents they hold is allowed; *i.e.* $1/N_P \leq \alpha \leq 1$. Thus, node p_i will hold a set of documents $\mathbf{D}_i \subseteq \mathbf{D}$, such that $|\mathbf{D}_i| = N_{D_i} = \alpha \cdot N_D$.

This type of document distribution reflects typical scenarios. In environments where nodes are under control of different entities (*e.g.* on the Web), overlap between the data is common. On the other hand, in environments where the complete network is under control of a single entity, specific data partitioning strategies can be achieved, including disjoint partitioning. Although in realistic situations the documents are not necessarily evenly distributed among nodes, we argue that this distribution is a reasonable approximation.

Document Categories

The original classification of the documents is assumed to be known, but not used during clustering, but rather during evaluation only. The set of categories, referred to here as *topics*[1], are represented as $T = \{t_c\}$, $c = 1, \ldots, N_T$. Each document is assumed to belong to one topic; *i.e.* $\text{Topic}(d_i) \in \mathbf{T}$.

4.2.2 Cluster Model

Upon clustering the documents, each node will have created a set of document clusters that best fit its local document collection. Thus, each node p_i maintains a set of clusters

[1]The term *topic* is used to refer to the document class to avoid confusion in mathematical notation.

$\mathbf{C}_i = \{c_r\}$, $r = 1, \ldots, N_{C_i}$. A cluster contains a subset of the documents in the node; *i.e.* c_r contains a subset of $\mathbf{D}_i$.

Clusters are allowed to overlap; *i.e.* each document can belong to more than one cluster, either in the same node or across different nodes. Thus, $\forall d_i$, it is possible that $\exists c_1, c_2 \in \mathbf{C}$, such that $d_k \in c_1$ and $d_k \in c_2$.

Document-to-cluster membership is represented as a relation $R(\mathbf{D}, \mathbf{C})$, which is a binary membership matrix:

$$\mathcal{M} = [m_{i,r}]$$

$$m_{i,r} = \left\{ \begin{array}{ll} 1 & \text{iff } d_i \in c_r \\ 0 & \text{otherwise} \end{array} \right\}$$

A projection of $\mathcal{M}$ over the documents dimension yields the number of documents in each cluster:

$$[\mathcal{M} \downarrow \mathbf{D}]_r = \sum_i m_{i,r}$$

Similarly, a projection of $\mathcal{M}$ over the clusters dimension yields the number of clusters to which each document belongs:

$$[\mathcal{M} \downarrow \mathbf{C}]_i = \sum_r m_{i,r}$$

In the process of distributed collaborative clustering, there will be a need to represent cluster summaries in the form of keyphrase vectors, so that they can be exchanged between peers. This is achieved using the CorePhrase algorithm introduced in chapter 3. The summary of cluster c_r is represented as a keyphrase vector λ_r, and is referred to as the cluster *core summary*. The set of cluster cores corresponding to the set of clusters at node

p_i is $\mathbf{\Lambda}_i = \{\lambda_r\}$, and will be referred to as the *node summary*. The keyphrases in each cluster core are the top L keyphrases extracted from the cluster using the CorePhrase algorithm described in section 4.3.2.

4.3 Collaborative Document Clustering Algorithm

The collaborative document clustering system relies on three components: an initial clustering algorithm using similarity histogram-based clustering (SHC), a cluster summarization algorithm (CorePhrase), and a distributed document clustering algorithm based on exchange of cluster summaries, recommendation and merging of peer documents. We discuss each of those components in the following subsections.

4.3.1 Initial Cluster Generation

Initial clustering is performed using a Similarity Histogram-based Clustering (SHC) [41]. The coherency of a cluster is represented as a **Cluster Similarity Histogram**.

> *Cluster Similarity Histogram*: A concise statistical representation of the set of pairwise document similarities distribution in the cluster. A number of *bins* in the histogram correspond to fixed similarity value intervals. Each bin contains the count of pair-wise document similarities in the corresponding interval.

Similarity. For the purpose of this work, we define the similarity between two documents as the ratio of their common features to the union of their features; i.e.

$$\text{sim}(d_i, d_j) = \frac{d_i \cap d_j}{d_i \cup d_j}$$

If each document is represented as a vector of keyword weights, we can calculate the similarity between a pair of documents using the widely used cosine coefficient:

$$\text{sim}(d_i, d_j) = \cos(d_i, d_j) = \frac{d_i \cdot d_j}{\|d_i\|\|d_j\|}$$

This cosine measure is used in our experiments to calculate document-to-document similarity. Regardless of which similarity function we choose, the similarity histogram concept remains neutral to our choice. The only requirement is that the similarity measure constitutes a metric on the document vector space.

Similarity Histogram Coherency. A coherent cluster should have high pairwise document similarities. A typical cluster has a normal distribution, while an ideal cluster would have a histogram where all similarities are maximum.

We judge the quality of a similarity histogram (cluster cohesiveness) by calculating the ratio of the count of similarities above a certain similarity threshold R_T to the total count of similarities. The higher this ratio, the more cohesive the cluster. Let N_{D_c} be the number of the documents in a cluster. The number of pair-wise similarities in the cluster is $N_{R_c} = N_{D_c}(N_{D_c}+1)/2$. Let $\mathbf{R} = \{r_i : i = 1, \ldots, N_{R_c}\}$ be the set of similarities in the cluster. The histogram of the similarities in the cluster is represented as:

$$H_c = \{h_i : 1 \leq i \leq B\} \tag{4.1a}$$

$$h_i = \text{count}(r_k), \qquad \delta \cdot (i-1) \leq r_k < \delta \cdot i \tag{4.1b}$$

where B : is the number of histogram bins,

h_i : is the count of similarities in bin i, and

δ : is the bin width of the histogram.

The histogram ratio (HR) of a cluster, which indicates cluster cohesiveness, is calculated as:

$$HR(c) = \frac{\sum_{i=T}^{B} h_i}{\sum_{j=1}^{B} h_j} \tag{4.2a}$$

$$T = \lfloor R_T \cdot B \rfloor \tag{4.2b}$$

where $HR(c)$: the histogram ratio of cluster c,

R_T : the similarity threshold, and

T : the bin number corresponding to the similarity threshold.

Similarity Histogram-based Clustering. The algorithm works by maintaining high HR for each cluster. New documents are tested against each cluster, adding them to appropriate clusters if they do not degrade the HR of that cluster significantly. Provisions are also made so as not to allow a chain reaction of "bad" documents being added to the same cluster, thus bringing its cohesiveness down significantly.

The algorithm (shown in Algorithm 4.1) works incrementally by iterating over the documents at node i, and for each cluster calculates the cluster histogram ratio before and after simulating the addition of the document to that cluster (lines 4-6). If the new ratio is greater than or equal to the old one, the document is added to the cluster. Otherwise if it is less than the old ratio by no more than ε and still above $HR_{\min}$, it is added (lines 7-9). Otherwise it is not added. If the document was not assigned to any cluster, a new cluster is created to which the document is added (lines 11-15).

4.3.2 Cluster Summarization Using Keyphrase Extraction

After initial clustering, a summary of each cluster c_r is computed as a set of core keyphrases, λ_r. The keyphrase extraction algorithm, CorePhrase, which is used in this stage of the

Algorithm 4.1 Similarity Histogram-based Incremental Document Clustering

```
1: C_i ← Empty List {Cluster List of Node i}
2: for each document d_j ∈ D_i do
3:    for each cluster c_r ∈ C_i do
4:       HR_old = HR(c_r)
5:       Simulate adding d_j to c_r
6:       HR_new = HR(c_r)
7:       if (HR_new ≥ HR_old) OR ((HR_new > HR_min) AND (HR_old − HR_new < ε)) then
8:          m_{j,r} ← 1 {Add d_j to c_r}
9:          Update HR(c_r)
10:      end if
11:   end for
12:   if [M ↓ C_i]_j = 0 {d_j was not added to any cluster} then
13:      Create a new cluster c_new
14:      C_i ← {C_i, c_new} {Add c_new to C_i}
15:      m_{j,new} ← 1 {Add d_j to c_new}
16:   end if
17: end for
```

algorithm, was discussed in chapter 3. Cluster summaries will be exchanged with peers to solicit their collaboration for enhancing the initial local clustering solution (described in section 4.3.3).

The usage of keyphrases to summarize document clusters is what enables peers to judge the similarity between their local data and the clusters of the node providing the summaries.

4.3.3 Collaborative Document Clustering

At this stage, each node p_i has an initial local clustering solution $\mathbf{C}_i$, and cluster summary information in the form of the core keyphrases of each cluster λ_r, which we will refer to as *cluster cores*. The set of cluster cores at a node is referred to as the node summary, $\mathbf{\Lambda}_i$

Algorithms 4.2 and 4.3 describe the collaborative document clustering process. Algo-

Algorithm 4.2 Recommend to Peers

1: **for** each peer p_j, $j = 1, \ldots, N_P$, $j \neq i$ **do**
Require: $\mathbf{C}_j$ and $\mathbf{\Lambda}_j$ have been calculated at peer p_j
2: Receive peer node summary $\mathbf{\Lambda}_j$
3: $\mathcal{S} = R(\mathbf{D}_i, \mathbf{\Lambda}_j) \leftarrow \emptyset$ {Similarity matrix between local data and peer summary}
4: **for** each document $d_k \in \mathbf{D}_i$ and cluster core $\lambda_r \in \mathbf{\Lambda}_j$ **do**
5: $\mathcal{S}(d_k, \lambda_r) \leftarrow (d_k \cap \lambda_r)/(d_k \cup \lambda_r)$
6: **end for**
7: **for** each cluster core $\lambda_r \in \mathbf{\Lambda}_j$ **do**
8: $D^+_{i,r} \leftarrow \{d_k \in \mathbf{D}_i \mid \mathcal{S}(d_k, \lambda_r) > R_T\}$
9: Send peer-positive set $D^+_{i,r}$ to peer p_j
10: Wait for peer p_j to merge peer-positive set $D^+_{i,r}$
11: Receive peer-positive set merge status $D^*_{i,r}$
12: **for** each $d^*_k \in D^*_{i,r}$ **do**
13: **if** status(d^*_k) == merged **then**
14: $c^*_i \leftarrow$ cluster to which d^*_k belongs
15: $HR_{\text{old}} = HR(c^*_i)$
16: Simulate removing d^*_k from c^*_i
17: $HR_{\text{new}} = HR(c^*_i)$
18: **if** $(HR_{\text{new}} > HR_{\text{old}})$ **then**
19: $m_{k,i^*} \leftarrow 0$ {Remove d^*_k from c^*_i}
20: **end if**
21: **end if**
22: **end for**
23: **end for**
24: **end for**

rithm 4.2 recommends documents to peers based on the received peer cluster summaries. It starts by exchanging cluster cores between peers. Each node receives $N_N - 1$ peer cluster summaries; each summary is a set of cluster cores $\mathbf{\Lambda}_j$. The receiving node compares the cluster cores to its own documents and builds a similarity matrix $\mathcal{S}$ between its local documents and peer cores. The document-to-cluster similarity is measured by calculating the ratio of common keywords between each document vector, $\mathbf{d}_k$, and the cluster keyphrase vector, λ_r, to the union of the two vectors (line 5).

Algorithm 4.3 Aggregate Peer Recommendation

for each peer p_j, $j = 1, \ldots, N_P$, $j \neq i$ **do**
  Receive recommendations $\{D_j^+\}$ from peer p_j
  for each recommendation $D_{j,r}^+ \in \{D_j^+\}$ **do**
    for each document $d_k^+ \in D_{j,r}^+$ **do**
      {Skip documents that already belong to this node}
      if $d_k^+ \ni \mathbf{D}_i$ **then**
        $c_{ir} \leftarrow$ cluster corresponding to recommendation $D_{j,r}^+$
        $HR_{\text{old}} = HR(c_{ir})$
        Simulate adding d_k^+ to c_{ir}
        $HR_{\text{new}} = HR(c_{ir})$
        if $(HR_{\text{new}} \geq HR_{\text{old}})$ OR $((HR_{\text{new}} > HR_{\text{min}})$ AND $(HR_{\text{old}} - HR_{\text{new}} < \varepsilon))$ **then**
          $m_{k,ir} \leftarrow 1$ {Add d_k^+ to c_{ir}}
          Update $HR(c_{ir})$
          status$(d_k^+) \leftarrow$ merged {set status of this document to merged}
        end if
      end if
    end for
  end for
  $\{D_j^*\} \leftarrow \{\text{status}(d_k) | d_k \in D_j^+\}$ {merge status vector}
  Send merge status vector $\{D_j^*\}$ to peer p_j
end for

A node then recommends to each peer those documents that are most similar (above a similarity threshold R_T) to the peer core summaries; we call these documents *peer-positive* documents. A peer-positive document does not necessarily have to leave the local cluster; *i.e.* it can be "copied" rather than "moved" to another peer. Peer-positive documents are sent as recommendation to the corresponding peer for evaluation and possible merging with its own clusters.

A peer decides what to do with the peer-positive recommendation set. Upon finishing, the peer returns a "merge status" vector indicating which of the peer-positive documents were merged with its own clusters. The node that receives that merge status vector then has

the opportunity to re-evaluate the membership of those merged peer-positive documents to its own clusters. If a merged peer-positive document is found to be affecting a local cluster negatively (by measuring the difference in histogram ratio before and after simulating the removal of the document), it is removed from the cluster. This strategy ensures that the local node never removes a document from its clusters unless it has been merged by another node.

Algorithm 4.3 aggregates the recommendations of each peer. It starts by receiving recommended documents from each peer, then follows the same logic of the SHC clustering algorithm described in section 4.3.1, except that it does not create new clusters for documents that were not assigned to any cluster. This decision was made to avoid creating small singleton clusters, which does not benefit the clustering solution. Instead, those documents are simply discarded, and their status will be "un-merged". When this peer reports back to the original node the merge status of the documents it received, the original node will know that those documents were not merged and thus will keep them regardless of their negative impact (if any) to its own clusters.

The algorithm promotes collaboration between peers by encouraging the exchange of documents so that everyone benefits. Documents in remote peers can be brought locally for merging if they are similar to a local cluster, and local negatively-contributing documents can be removed if they can be merged with remote clusters. This process tends to create better clusters at each node. This is confirmed through experimental evaluation where the final clusters are always of higher quality than the initial clusters.

4.3.4 Communication Complexity

We can estimate the number of messages passed between peers by analyzing the algorithms where sending or receiving information is needed. Each peer sends to all other peers

summarized clusters, equal to $N_P(N_P-1)/2$ messages. If we denote the maximum number of clusters at any node by K, then we have $O(KN_P^2)$ messages being sent for the summary broadcast part. Recommendation messages consist of a set of documents, $D_i^+ \subset D_i$, from each peer to all other peers. D_i^+ depends on the similarity threshold, R_T. If we denote the maximum set of recommended documents from any peer as D^+, then we have $O(D^+N_P^2)$ messages for recommendation. Merge status messages will be even smaller in size than recommendation messages, so we can neglect them. Then total communication complexity will be $O((K + D^+)N_P^2)$.

4.4 Summary

In this chapter a method for locally-optimized distributed clustering was presented. The method employs collaborative clustering between distributed nodes, through the exchange of cluster summaries. By giving remote peers a summary of local clusters, a node allows its peers to judge the similarity between their local data and the node's clusters. This in turn allows the peers to make recommendation of which data objects is most likely to enhance the quality of the node's data. Data can be copied or moved from one node to another based on this collaboration, which results in the improvement in the local clustering quality.

This process implies that clustering under this collaboration scheme will produce asymmetric clustering solutions, hence the locally-optimized clustering. It is suitable for scenarios where clusters are preferred to be maintained wholly by certain nodes, as opposed to the globally-optimized clusters, where one set of cluster prototypes is computed over all nodes.

Experimental results in chapter 6, section 6.3, illustrate the improvement achieved in clustering quality through node collaboration, showing a significant improvement in

final local clustering over the initial clustering before collaboration. Statistical significance testing is performed to back this claim. The effect of data distribution ratio is also tested, and shows that that as more data overlap exists, collaboration produces less improvement in clustering quality.

In the next chapter we introduce a globally-optimized distributed clustering method, called hierarchically-distributed peer-to-peer clustering (HP2PC), which aims for scalability and modularity of the distributed clustering architecture and algorithm. Unlike locally-optimized distributed clustering, the HP2PC method produces a single set of globally-optimized cluster prototypes across the whole network.

CHAPTER 5

Hierarchically-Distributed Peer-to-Peer Document Clustering

In distributed data mining models, adopting a flat node distribution model can affect scalability. To address the problem of modularity, flexibility and scalability, we propose a Hierarchically-distributed Peer-to-Peer Clustering (HP2PC) architecture and algorithm [48]. The architecture is based on a multi-layer overlay network of peer neighborhoods. Supernodes, which act as representatives of neighborhoods, are recursively grouped to form higher level neighborhoods. Peers at a certain level of the hierarchy form the overlay network at that level, and cooperate within their respective neighborhoods to perform P2P clustering. Using this model, we can partition the clustering problem in a modular way among neighborhoods, solve each part individually, then successively combine clusterings up the hierarchy where increasingly more global solutions are computed. We applied the model to a distributed document clustering problem and achieved decent speedup (reaching 155 times faster than centralized clustering, for a 250-node network)

with comparable clustering quality to the centralized approach.

5.1 Overview

A recent shift towards distributed data mining (DDM) was sparked by the data mining community since the mid 90s. It was realized that analyzing massive data sets, that often span different sites, using traditional centralized approaches can be intractable. In addition, DDM is being fueled by recent advances in grid infrastructures and distributed computing platforms.

Huge data sets are being collected daily in different fields; *e.g.* retail chains, banking, biomedicine, astronomy, *etc.*, but it is still extremely difficult to draw conclusions or make decisions based on the collective characteristics of such disparate data.

Four main approaches for performing DDM can be identified. A common approach is to bring the data to a central site, then apply centralized data mining on the collected data. Such an approach clearly suffers from a huge communication and computation cost to pool and mine the global data. In addition, it cannot preserve data privacy.

A smarter approach is to perform local mining at each site to produce a local model. All local models can then be transmitted to a central site that combines them into a global model [93, 83, 22]. Ensemble methods also fall into this category [97]. While this approach may not scale well with the number of nodes, it can be considered better than pooling the data.

Another smart approach is for each site to carefully select a small set of representative data objects and transmit them to a central site, which combines the local representatives into one global representative data set. Data mining can then be carried on the global representative data set [60, 67].

All three previous approaches involve a central site to facilitate the DDM process. A more departing approach does not involve centralized operation, and thus belongs to the peer-to-peer (P2P) class of algorithms. P2P networks can be unstructured or structured. Unstructured networks are formed arbitrarily by establishing and dropping links over time, and they usually suffer from flooding of traffic to resolve certain requests. Structured networks, on the other hand, make an assumption about the network topology and implement a certain protocol that exploits such a topology.

In P2P DDM, sites communicate directly with each other to perform the data mining task [32, 24, 25, 23]. Communication in P2P DDM can be very costly if care is not taken to localize traffic, instead of relying on *flooding* of control or data messages.

In this chapter we introduce an approach for distributed data clustering, based on a structured P2P network architecture. The goal is to achieve modularity, flexibility and scalability. The proposed model is called Hierarchically-distributed Peer-to-Peer Clustering (HP2PC). It involves a hierarchy of P2P neighborhoods, in which the peers in each neighborhood are responsible for building a clustering solution, using P2P communication, based on the data they have access to. As we move up the hierarchy, clusters are merged from lower levels in the hierarchy. At the root of the hierarchy one global clustering can be derived.

The model deviates from the standard definition of P2P networks, which typically involve loose structure (or no structure at all), based on peer connections that are created and dropped frequently. The HP2PC model, on the other hand, is based on static hierarchical structure that is designed up front, upon which the peer network is formed. We plan to introduce a dynamic structure extension to this model in future work.

Using the HP2PC model, we can partition the problem in a modular way, solve each part individually, then successively combine solutions if it is desired to find a global solu-

tion. This way, we avoid two problems in the current state-of-the-art DDM: (a) we avoid high communication cost usually associated with a structured, fully-connected network, and (b) we avoid uncertainty in the network topology usually introduced by unstructured P2P networks. Experiments performed on document clustering show that we can achieve comparable results to centralized clustering with high gain in speedup.

The model lends itself to real-world structures, such as hierarchically distributed organizations or government agencies. In such scenario, different departments or branches can perform local clustering to draw conclusions from local data. Parent departments or organizations can combine results from those in lower levels to draw conclusions on a more holistic view of the data.

The following section introduces the HP2P distributed architecture. Section 5.3 discusses the foundation behind the HP2P distributed clustering algorithm, and discusses its complexity. Finally, a summary is given in section 5.6.

5.2 The HP2PC Distributed Architecture

HP2PC is a hierarchically-distributed P2P architecture for scalable distributed clustering. We argue that a scalable distributed clustering system (or any data mining system for that matter) should involve hierarchical distribution. A hierarchical processing strategy allows for delegation of responsibility and modularity.

Central to this hierarchical architecture design is the formation of *neighborhoods*. A neighborhood is a group of peers forming a logical unit of isolation in an otherwise unrestricted open P2P network. Peers in a neighborhood can communicate directly, but not with peers in other neighborhoods. Each neighborhood has a *supernode*. Communication between neighborhoods is achieved through their respective supernodes. This model

reduces flooding problems usually encountered in large P2P networks.

The notion of a neighborhood accompanied by a supernode can be applied recursively to construct a multi-level overlay hierarchy of peers; *i.e.* a group of supernodes can form a higher level neighborhood, which can communicate with other neighborhoods on the same level of the hierarchy through their respective (higher-level) supernodes. This type of hierarchy is illustrated in figure 5.1.

5.2.1 Hierarchical Overlays

A P2P network is comprised of a set of peers, or nodes, $\mathbf{P} = \{p_i\}_1^{N_P}$. An *overlay* network is a logical network on top of $\mathbf{P}$ that connects a certain subset of the nodes in $\mathbf{P}$. Most work in the P2P literature refers to a single level of overlay; *i.e.* there exists one overlay network on top of the original P2P network. In our work, this concept is extended further to allow multiple overlay levels on top of $\mathbf{P}$.

To distinguish between the different levels of overlay, we will use the *height* of an overlay network. An overlay network at height h is denoted $\mathbf{P}^{(h)}$. The lowest level network (the original P2P network) is at height $h = 0$, thus denoted $\mathbf{P}^{(0)}$; while the highest overlay possible is at height H, denoted $\mathbf{P}^{(H)}$, and consists of a single node (the *root* of the overlay hierarchy). In subsequent formulations, we will drop the superscript (h) if the formulation is referring to an arbitrary level that does not require level information; otherwise we will use it to distinguish overlay levels.

The size of the overlay network at each level will differ according to how many peers comprise the overlay. However, since a higher level overlay will always be a subset of its immediate lower level overlay, we maintain the following inequality:

$$0 < |\mathbf{P}^{(h)}| < |\mathbf{P}^{(h-1)}|, \forall h > 0. \tag{5.1}$$

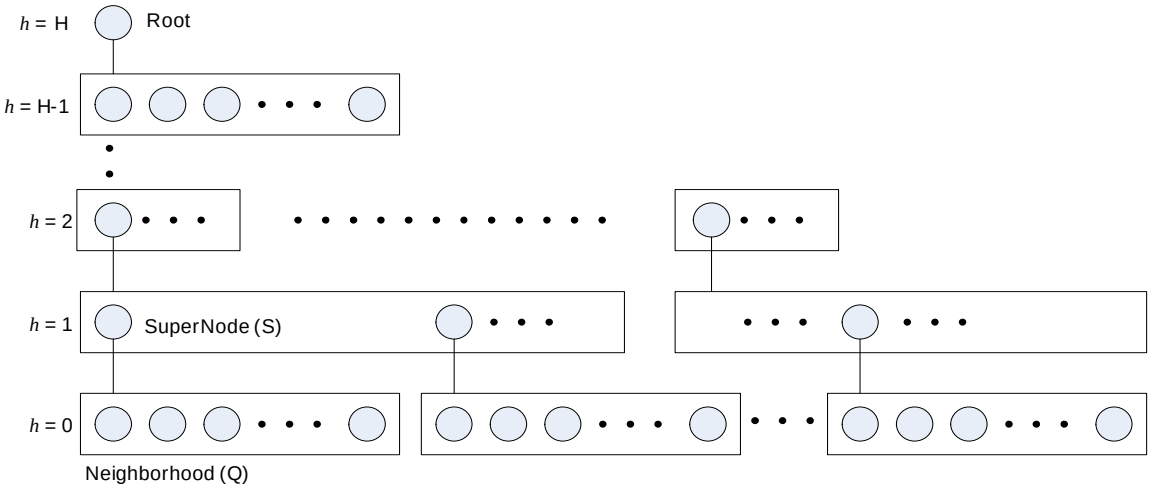


Figure 5.1: The HP2PC Hierarchy Architecture

The choice of which subset of peers forms the next level overlay is closely related to the next subsection discussing peer neighborhoods.

5.2.2 Neighborhoods

A neighborhood, Q, is a set of peers that is subset of an overlay $\mathbf{P}$, and that has a designated peer known as a *supernode*, p_s; thus

$$\text{Neighborhood} \equiv (Q, p_s) : Q \subseteq \mathbf{P}, p_s \in Q.$$

The following neighborhood properties are enforced for the HP2PC architecture:

- A set of neighborhoods, $\mathbf{Q} = \{Q_j\}_1^{N_Q}$, covers the overlay network $\mathbf{P}$:

$$\mathbf{P} \equiv \bigcup_{j=1}^{N_Q} Q_j.$$

- Neighborhoods do not overlap:

$$\forall i, j \neq i : Q_i \cap Q_j = \emptyset.$$

- A node must belong to some neighborhood:

$$\forall p \in \mathbf{P} : p \in \text{some } Q.$$

A network partitioning factor, $\beta \in \mathbb{R}$, $0 \leq \beta \leq 1$, partitions the P2P network into equally sized neighborhoods. The number of neighborhoods in an overlay network as a function of β is then given by

$$N_Q = \lfloor 1 + \beta(N_P - 1) \rfloor \tag{5.2}$$

At one extreme, when $\beta = 0$, there is only one neighborhood that contains all the peers in $\mathbf{P}$. On the other extreme, when $\beta = 1$, there are N_P neighborhoods, each containing one peer only. In between we can set β to a value that determines the number of neighborhoods, and consequently the size of each neighborhood.

An initial attempt to determine the size of each neighborhood as a fraction of the size of the network was to allocate for each neighborhood a floor-rounded equal partition of the nodes, and then allocate the remaining nodes to the last neighborhood; *i.e.*

$$\text{size}_j = \begin{cases} \lfloor N_P / N_Q \rfloor & 1 \leq j \leq (N_Q - 1) \\ N_P - \sum_{k=1}^{N_Q - 1} \text{size}_k & j = N_Q \end{cases} \tag{5.3}$$

However, this only works well when $N_P \gg N_Q$. When N_P and N_Q are of the same order of magnitude, size_j tends to be a small real number. The problem is that such small

number will cause all neighborhoods, except the N_Q^{th}, to have a very small number of nodes, while the N_Q^{th} will absorb the remaining (large number of) nodes. For example, if N_P is 250, and N_Q is 150, then the we have 149 neighborhoods with size 1, and one neighborhood with size 101.

To solve this problem, we use a binary function to generate the neighborhood sizes based on a specified probability for each value. Let r represent this probability, given by:

$$r = (N_P/N_Q) - \lfloor N_P/N_Q \rfloor.$$

The following function then generates two neighborhood sizes based on r:

$$\text{size} = \begin{cases} \lfloor N_P/N_Q \rfloor & \text{with probability } (1-r) \\ \lceil N_P/N_Q \rceil & \text{with probability } r \end{cases} \tag{5.4}$$

This method produces a more even distribution of neighborhood sizes, even if N_P and N_Q are of the same order of magnitude. In the example above, using equation 5.4 we have $N_P/N_Q = 1.67$ ($r = 0.67$), so 67% of the neighborhoods will be of size 2, and 33% will be of size 1, far more balanced than with equation 5.3.

All peers on the same level, h, of the hierarchy are denoted by $p^{(h)}$. Let the function level(p) determine the level of a peer; *i.e.* level($p^{(h)}$) $= h$. A peer p_i can communicate with peer p_j if, and only if, level(p_i) = level(p_j) and $p_i \in Q \iff p_j \in Q$.

Peer hierarchy formation is bottom-up, so the lowest level of the hierarchy is $h = 0$. The supernodes of level 0 neighborhoods form the overlay network at level $h = 1$. Recursively, at level $h = 2$ are the supernodes of level 1 neighborhoods (groups of level 1 supernodes.) The root supernode is at level H, the height of the hierarchy; *i.e.* there exists exactly one $p^{(H)}$ in the system.

The network partitioning factor, β, can be different for different levels of the hierarchy; *i.e.* neighborhood count and size is not necessarily the same at each level. If we apply the same network partitioning factor to every level we can deduce the height, H, of the hierarchy. We can approximate equation 5.2 to

$$N_Q \approx \beta N_P$$

Since the number of nodes at a certain level is equal to the number of superndoes (or neighborhoods) in the lower level, then we can say that

$$N_P^{(h)} = \beta N_P^{(h-1)}, \forall h > 0$$

At the top of the hierarchy (level H) we have one node, then

$$\beta^H N_P^{(0)} = 1 \tag{5.5}$$

from which we can deduce H:

$$H = \begin{cases} \lceil -\frac{\log N_P^{(0)}}{\log \beta} \rceil & 0 < \beta < 1 \\ 1 & \beta = 0 \\ \text{undefined} & \beta = 1 \end{cases} \tag{5.6}$$

If, however, the partitioning factor is chosen to be different for different levels of the hierarchy, then we cannot deduce the full height of the hierarchy up front. However, we can deduce the maximum height reachable from a certain level using equation 5.6, and hence we can iteratively calculate the full hierarchy height if all level βs are known a priori.

If, instead of specifying βs, a certain hierarchy height is desired (*e.g.* to mirror an

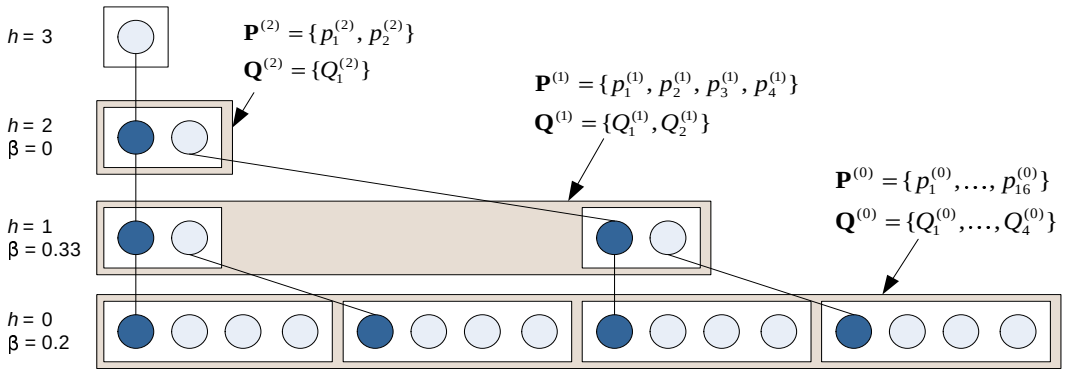


Figure 5.2: Example of HP2PC Network

already existing hierarchical structure), we can calculate the proper β (the same for all levels) using the following equation (which is derived from equation 5.6):

$$\beta = \begin{cases} e^{-(\log N_P^{(0)})/H} & H > 1 \\ 0 & H = 1 \end{cases} \tag{5.7}$$

5.2.3 Example

Figure 5.2 illustrates the HP2PC architecture with an example. The network shown consists of 16 nodes and 4 hierarchy levels. The set of nodes at level 0, $\mathbf{P}^{(0)}$, are divided into 4 neighborhoods, subject to the network partitioning factor $\beta^{(0)} = 0.2$. Each supernode of level 0 becomes a regular node at level 1, forming the set of 4 nodes $\mathbf{P}^{(1)}$. Those in turn are grouped into two neighborhoods forming $\mathbf{Q}^{(1)}$, satisfying $\beta^{(1)} = 0.33$. At level 2, only one neighborhood is formed out of level 1 supernodes, satisfying $\beta^{(1)} = 0$. Finally the root of the hierarchy is found at level 3.

Algorithm 5.1 HP2PC Construction

Require: $\mathbf{P}^{(0)}$, $\beta^{(h)}$ $(1 \leq h \leq H-1)$
  $\mathbf{P}$ = ConstructHP2PC($\mathbf{P}^{(0)}, \beta[], 0$)
Function ConstructHP2PC($\mathbf{P}, \beta[], h$)
1: **if** $|\mathbf{P}^{(h)}| = 1$ **then**
2: **return** $\mathbf{P}$ {stopping criterion}
3: **end if**
4: Calculate $N_Q^{(h)}$ by substituting $|\mathbf{P}^{(h)}|$ and $\beta[h]$ into equation 5.2
5: $\mathbf{Q}^{(h)} = \phi$, $\mathbf{P}^{(h+1)} = \phi$
6: $m = 1$, $n = 1$ {Partition $\mathbf{P}$ into N_Q neighborhoods}
7: **for** $j = 1$ to N_Q **do**
8: Calculate size$_j$ using equation 5.4
9: $n = n + \text{size}_j$
10: $Q_j = \mathbf{P}^{(h)}[m:n]$
11: $m = n + 1$
12: Add Q_j to $\mathbf{Q}^{(h)}$
13: Designate $Q_j[1]$ as supernode for Q_j
14: Add $Q_j[1]$ to $\mathbf{P}^{(h+1)}$
15: **end for**
16: $\mathbf{P}$ = ConstructHP2PC($\mathbf{P}, \beta[], h+1$) {recursively build higher levels}
17: **return** $\mathbf{P}$

5.2.4 HP2PC Network Construction

An HP2PC network is constructed recursively, starting from level 0 up to the height of the hierarchy, H. The number of neighborhoods and size of each neighborhood, is controlled through the partitioning factor β which is specified for each level of the hierarchy (except the root level).

The construction process is given in algorithm 5.1. Given the initial set of nodes, $\mathbf{P}^{(0)}$, and the set of partitioning factors $\beta[]$, the algorithm recursively constructs the network. At each level we partition the current $\mathbf{P}^{(h)}$ into the proper number of neighborhoods, and assign a supernode for each one. The set of supernodes at a certain level form the set of

nodes for the next higher level, which are passed to the next recursive call. Construction stops when the root is reached.

5.3 The HP2PC Distributed Clustering Algorithm

The HP2PC algorithm is a distributed iterative clustering process. It is a centroid-based clustering algorithm, where a set of cluster centroids are generated to describe the clustering solution. In HP2PC, each neighborhood converges to a set of centroids that describe the data set in that neighborhood. Other neighborhoods, either on the same level or at higher levels of the hierarchy, may converge to another set of centroids.

Once a neighborhood converges to a set of centroids, those centroids are acquired by the supernode of that neighborhood. The supernode, in turn as part of its higher level neighborhood, collaborates with its peers to form a set of centroids for its neighborhood. This process continues hierarchically until a set of centroids are generated at the root of the hierarchy.

5.3.1 Estimating Clustering Quality

The distributed search for cluster centroids is guided by a cluster quality measure that estimates intra-cluster cohesiveness and inter-cluster separation. The measure is similar to the Histogram Ratio introduced in chapter 4, but instead of working with the histogram ratio, we calculate the first moment of the histogram; *i.e.* its skew.

Cluster Cohesiveness. The distribution of pair-wise similarities within a cluster is represented using a *cluster similarity histogram*, which is a concise statistical representation of the cluster tightness [41].

Let $\text{sim}(\cdot)$ be a similarity measure between two objects, and S_c be the set of pair-wise

similarity between objects of cluster c:

$$S_c = \{s_k : 1 \leq k \leq |c|(|c|+1)/2\} \tag{5.8a}$$

$$s_k = \text{sim}(d_i, d_j), \quad d_i, d_j \in c \tag{5.8b}$$

where $|c|$ is the number of the objects in a cluster. The histogram of the similarities in the cluster is represented as:

$$H_c = \{h_i : 1 \leq i \leq B\} \tag{5.9a}$$

$$h_i = \text{count}(s_k), \tag{5.9b}$$

$$s_k \in S_c,\ \delta \cdot (i-1) \leq s_k < \delta \cdot (i) \tag{5.9c}$$

where B : the number of histogram bins,

h_i : the count of similarities in bin i, and

δ : the bin width of the histogram.

To estimate the cohesiveness of cluster c, we calculate the histogram *skew*. Skew is the third central moment of a distribution; it tells us if one tail of the distribution is longer than the other. A positive skew indicates a longer tail in the positive direction (higher interval of the histogram), while a negative skew indicates a longer tail in the negative (lower interval) direction. A similarity histogram that is negatively skewed indicates a tight cluster.

Skew is calculated as

$$\text{skew} = \frac{\sum_i (x_i - \mu)^3}{N\sigma^3} \tag{5.10}$$

so, the tightness of a cluster, $\varphi(c)$, calculated as the skew of its histogram H_c, is

$$\varphi(c) = \text{skew}(H_c) = \frac{\sum_k (s_k - \mu_{S_c})^3}{|S_c|\sigma_{S_c}^3}, \quad s_k \in S_c. \tag{5.11}$$

A clustering quality measure based on skewness of similarity histograms of individual clusters can be derived as a weighted average of the individual clusters skew:

$$\varphi(C) = \frac{\sum_i |c_i|\varphi(c_i)}{\sum_i |c_i|}, \quad c_i \in C. \tag{5.12}$$

5.3.2 Distributed Clustering (Level $h = 0$)

We define a general function for updating cluster models in a fully-connected neighborhood:

$$C_i^t = f(\{C_j^{t-1}\}), \quad i, j \in Q \tag{5.13a}$$

$$C_i^0 = C^0 \tag{5.13b}$$

where C_i^t is the clustering model (usually a set of centroids) calculated by peer i at iteration t, and $f(\cdot)$ is an aggregating function. The equation can be illustrated by figure 5.3, where the output of each peer at iteration t depends on the models calculated by all other peers at iteration $t-1$. In P2P k-means [24], $f(\cdot) \equiv \text{avg}(\cdot)$, and the neighborhood is based on network topology.

The HP2PC algorithm iteratively updates the cluster centroids at each node. For each neighborhood an initial set of centroids, m^0, is calculated by the supernode and transmitted to all peers in the neighborhood. Like k-means, during each iteration each peer assigns

Algorithm 5.2 Level 0 Neighborhood Clustering

Require: Neighborhood $Q = \{p_i\}$, #clusters K
Ensure: C
1: $S = \text{CalcSimilarityMatrix}(D_i)$
2: $\{(m_s, w_s)\} = \text{ReceiveFrom}(p_s)$ $\{p_s$: supernode of $Q\}$
3: $\forall j \neq s : \{(m_j, w_j)\} = 0$
4: $\{c, \varphi\} = \text{UpdateClusters}(D_i, \{m_i\})$
5: **while** change in $\{m_i\} > \epsilon$ **do**
6: **for all** $p_j \in Q, j \neq i$ **do**
7: $\text{SendTo}(p_j, \{(m_i, w_i)\})$
8: $\{(m_j, w_j)\} = \text{ReceiveFrom}(p_j)$
9: **end for**
10: **for all** $k \in [1, K]$ **do**
11: $m_{ik} = 0, w_k = 0$
12: **for all** $j \in Q$ **do**
13: $m_{ik} = m_{ik} + (m_{jk} \cdot w_{jk})$
14: $w_k = w_k + w_{jk}$
15: **end for**
16: $m_{ik} = m_{ik}/w$
17: **end for**
18: $\{c, \varphi\} = \text{UpdateClusters}(D_i, \{m_i\})$
19: **end while**

Algorithm 5.3 Utility Routines for Neighborhood Clustering

Function UpdateClusters($D, \{m\}$)

1: $c = \{\}$
2: **for all** $d \in D$ **do**
3: $l = \text{argmin}_k\{|d - m_k|\}$
4: $c_l \leftarrow \{c_l, d\}$
5: **end for**
6: **for all** c_k **do**
7: $m_k = \text{avg}(d_i)_{i \in c_k}$
8: $S_k = S \downarrow c_k$ {part of S indexed by objects in c_k}
9: $\varphi_k = \text{CalcSkew}(S_k)$
10: **end for**
11: **return** $\{c, \varphi\}$

Function CalcSkew(S)

1: $\varphi \leftarrow 0$, $\mu \leftarrow \text{avg}(S)$, $\sigma \leftarrow \text{stddev}(S)$
2: **for all** $s_k \in S$ **do**
3: $\varphi = \varphi + (s_k - \mu)^3$
4: **end for**
5: $\varphi = \varphi / (|S| \times \sigma^3)$
6: **return** φ

Function CalcSimilarityMatrix(D)

1: $S \leftarrow \phi$
2: **for** $i = 1$ to $|D|$ **do**
3: **for** $j = 0$ to $i - 1$ **do**
4: $S \leftarrow \{S, \text{sim}(D[i], D[j])\}$
5: **end for**
6: **end for**
7: **return** S

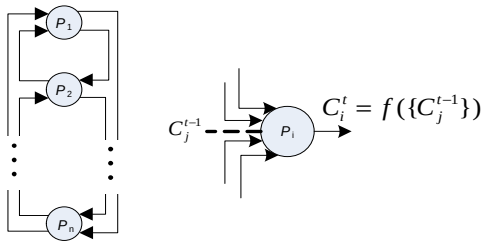


Figure 5.3: Distributed Clustering Iterative Loop

local data to their nearest centroid, and calculates their new centroids. In addition, it also calculates cluster skews using equation 5.11.

The final set of centroids for each iteration is calculated from all peer centroids. Unlike k-means (or P2P k-means), those final centroids are weighed by the skew and size of clusters at individual peers. At the end of each iteration, peers broadcast cluster centroid, skew, and size information:

$$C_j^t \equiv (m_{jk}, \varphi_{jk}, |c_{jk}|)^t, \quad k \in [1, K]$$

The weight of a cluster k at peer j is defined as:

$$w_{jk} = \varphi(c_{jk}) \cdot |c_{jk}|$$

At peer i, the centroid of cluster k is updated according to the following equation, which favors tight and dense clusters:

$$m_{ik}^t = \frac{\sum_j w_{jk}^{t-1} \cdot m_{jk}^{t-1}}{\sum_j w_{jk}^{t-1}}, \quad j \in Q \tag{5.14}$$

This is followed by assigning objects to their nearest centroid, and calculating the new set of cluster skews, φ_{ik}, and sizes, $|c_{ik}|$, which are used in the next iteration. The algorithm

terminates when object assignment does not change, or when $\forall i, k \quad |m_{ik}^{t} - m_{ik}^{t-1}| < \epsilon$, where ϵ is a sufficiently small parameter.

5.3.3 Distributed Clustering (Level $h > 0$)

Once a neighborhood converges to a set of clusters, the centroids and weights of those clusters are acquired by the supernode as its initial set of clusters; *i.e.* for neighborhood Q with supernode p_s

$$C_s^{(0,h)} = C_Q^{(T,h-1)}$$

where T is the final iteration of the algorithm at level $h-1$ for neighborhood Q.

Since at level h of the hierarchy the actual data objects are not available, we rely on *meta-clustering*: merging the clusters using centroid and weight information alone. At level $h > 0$, clusters are merged in a bottom-up fashion, up to the root of the hierarchy; *i.e.* $C^{(h)} = f(C^{(h-1)})$. This means once a neighborhood at level h converges to a set of clusters, it is frozen, and the higher level clustering is invoked. (A more elaborate technique would involve bi-directional traffic, making $C^{(h-1)} = f(C^{(h)})$ as well, but the complexity of this approach could be prohibitive, so we leave it for future work.)

A neighborhood at level h consists of a set of peers, each having a set of K centroids. To merge those clusters, the centroids are collected and clustered at the supernode of this neighborhood, using k-means clustering. This process repeats until one set of clusters is computed at the root of the hierarchy. The formal procedure representing this clustering process is presented in algorithm 5.4.

One of the major benefits of this algorithm is the ability to *zoom in* to more refined clusters by descending down the hierarchy, and *zoom out* to more generalized clusters

Algorithm 5.4 HP2PC Clustering

```
1: for all Q_i ∈ Neighborhoods(h = 0) do
2:    {m_i}^(0) = NeighborhoodCluster(Q_i)
3: end for
4: for h = 1 to H do
5:    for all Q_i ∈ Neighborhoods(h) do
6:       for all p_j ∈ Q_i do
7:          {m_j} = {m_j}^(h−1)
8:          SendTo(p_s, {m_j})
9:       end for
10:      {m_i}^h = kmeans({m_j}) {only at peer p_s}
11:   end for
12: end for
```

by ascending up the hierarchy. The other major benefit is the ability to merge a forest of independent hierarchies into one hierarchy by putting all roots of the forest into one neighborhood and invoking the merge algorithm on that neighborhood.

5.4 Distributed Cluster Summarization

Summarizing the clusters generated by HP2PC using CorePhrase poses two challenges. First, since CorePhrase works by intersecting documents in a cluster together, generating a summary for a document cluster that is distributed across various nodes cannot be done directly, and thus CorePhrase needs modification to work in this kind of environment. Second, merging cluster summaries up the hierarchy will require working with keyphrases extracted at level 0 only, without any access to the actual documents.

5.4.1 Summarizing Level 0 Clusters

Let every node p_i generate a cluster summary λ_{ik} (set of keyphrases) for each cluster c_k using CorePhrase. Nodes in the same neighborhood will enter into multiple rounds of communication to agree on a common summary for each cluster. For cluster c_k, on each round, node p_i receives a cluster summary $\lambda_{jk}, j \in Q$, from all nodes in its neighborhood. Node p_i then produces two sets of keyphrases based on λ_{jk}, one is called *core summary*, λ_{Qk}, and the other is called *local cluster summary*, λ_{ik}. The core summary is generated by intersecting all keyphrases in $\{\lambda_{jk}, \lambda_{ik}\}$:

$$\lambda_{Qk} = \bigcap_{j \in Q} \lambda_{jk} \tag{5.15}$$

The local cluster summary is generated by intersecting all summaries from other nodes with local documents from cluster c_k:

$$\lambda_{ik} = \bigcup_{j \in Q, j \neq i} \lambda_{jk} \cap \mathbf{D}_{ik} \tag{5.16}$$

Note that, by definition, the core summary will be the same at all nodes, since the operation is identical at all nodes. The local cluster summary, however, will be different due to intersection with local documents. On the next iteration, each node will send its local cluster summary to all other nodes. Local cluster summaries are intersected together again according to equation 5.15, and the result is appended to the core summary λ_{Qk}:

$$\lambda_{Qk}^{(t+1)} = \lambda_{Qk}^{(t)} \cup \left[\bigcap_{j \in Q} \lambda_{jk}^{(t)} \right] \tag{5.17}$$

This process repeats until the desired number of keyphrases per cluster summary is

acquired, or the intersection yields an empty set.

5.4.2 Summarizing Higher Level Clusters

At higher levels, equation 5.16 is not applicable, since no local data is available. Summarization of a cluster at higher levels is simply an intersection of keyphrase summaries of the clusters chosen to be merged into said cluster by the higher level k-means algorithm. Let $C_k = \{c_i\}$ be the set of clusters chosen to be merged into one cluster, c_k, and let $\{\lambda_i\}$ be their corresponding summaries. The summary for cluster c_k is the intersection of the constituent cluster summaries, merged with the an equal subset from every constituent summary, up to L keyphrases.

Let λ_k represent the core intersection:

$$\lambda_k = \bigcap_{i \in C_k} \lambda_i$$

If $|\lambda_k| > L$, then λ_k is truncated up to L keyphrases. Otherwise, λ_k is merged with an equal subset from every constituent cluster summary λ_{ki}. Let $M = (L - |\lambda_k|)/K$, where K is the number of clusters; and let $\lambda^*_{ik} = \lambda_{ik} \setminus \lambda_k$ then the final cluster summary is:

$$\lambda_k \leftarrow \{\lambda_k, \bigcup_i \lambda^{*(M)}_{ik}\} \tag{5.18}$$

where $\lambda^{*(M)}_{ik}$ denotes the top M keyphrases in λ^*_{ik}. Thus, the core cluster summary is augmented with an equal subset from the top keyphrases in each constituent cluster summary that is not in the core summary. This is to make sure that the core summary is representative of all constituent clusters.

5.5 Complexity Analysis

We divide the complexity of HP2PC into computational complexity and communication complexity.

5.5.1 Computation Complexity

Assume the whole data set size for all nodes is D. Data is equally partitioned among nodes, so each node holds $D_P = D/N_P$ data objects. For level 0, we have N_Q neighborhoods, each of size $S_Q = N_P/N_Q$.

Each node has to compute a pair-wise similarity matrix before it begins the P2P clustering process, requiring $D_P(D_P - 1)/2$ similarity computations. For each iteration, each node computes a new set of K centroids by averaging all neighborhood centroids ($K \cdot S_Q$), assigns the data objects to those centroids ($K \cdot D_P$), recomputes centroids based on new data assignment (D_P) , and calculates the skew of the clusters ($D_P(D_P - 1)/2$). Those requirements are summarized as:

$$\begin{aligned} T_{\text{sim}} &= D_P(D_P - 1)/2 \\ T_{\text{update}} &= K(S_Q^{(0)} + D_P) + D_P \\ T_{\text{skew}} &= D_P(D_P - 1)/2 \end{aligned}$$

Let the number of iterations required to converge to a solution be I. Then the total number of computations required by each node to converge is:

$$T_P = T_{\text{sim}} + I\,[T_{\text{update}} + T_{\text{skew}}] \tag{5.19}$$

If we assume that $D_P \gg 1$ and $I \gg 1$, then we can rewrite equation 5.19 as:

$$T_P = I(K(S_Q^{(0)} + D_P) + D_P^2/2) \tag{5.20}$$

For levels above 0, each neighborhood is responsible for meta-clustering a set of KS_Q centroids into K centroids using k-means. Then for each neighborhood at level h, the required computation is:

$$T_h = IS_Q^{(h)}K^2 \tag{5.21}$$

Since each neighborhood computation is done in parallel with the others, we need only T_h computations per level. However, since computations at higher levels of the hierarchy need to wait for lower levels to complete, we have to sum T_h for all levels. The total computations required for all levels above 0 are thus:

$$T_H = \sum_{h=1}^{H-1} T_h \tag{5.22}$$

Finally, we can combine equations 5.20 and 5.22 to find the total computation complexity for HP2PC:

$$T = T_P + T_H \tag{5.23}$$

It can be seen that computation complexity is largely affected by the data set size of each node (D_P). By increasing the total number of nodes we can decrease D_P (since the data is equally partitioned among nodes), but at the expense of increasing communication complexity, as well as decreasing clustering quality due to fragmentation of the data set.

5.5.2 Communication Complexity

In neighborhoods at level 0, at each iteration every peer sends $S_Q - 1$ messages to its neighbors, and each message is of size K. For S_Q peers in one neighborhood, $S_Q(S_Q - 1)$ messages are exchanged. Communication complexity for $N_Q^{(0)}$ neighborhoods at level 0 is then:

$$M_0 \approx N_Q^{(0)} S_Q^{(0)2} IK \tag{5.24}$$

Since $N_Q = N_P/S_Q$, then:

$$M_0 \approx N_P^{(0)} S_Q^{(0)} IK \tag{5.25}$$

For levels above 0, each neighborhood requires $S_Q - 1$ messages to be sent to each supernode, and each message is of size K. Communication complexity for $N_Q^{(h)}$ neighborhoods at level 0 is then:

$$\begin{aligned} M_h &\approx N_Q^{(h)} S_Q^{(h)} K \\ &\approx N_P^{(h)} K \end{aligned} \tag{5.26}$$

Total communication requirements for HP2PC is then:

$$M \approx M_0 + \sum_{h=1}^{H-1} M_h \tag{5.27}$$

We can see that the communication complexity is greatly influenced by the size of neighborhoods at level 0. The worst case is when all nodes are put into one neighborhood, resulting in quadratic complexity (in terms of the number of nodes). As we adopt more fine grained neighborhoods we can reduce both computation and communication complexity,

but at the expense of clustering quality as will be discussed in the results section.

5.6 Summary

In this chapter a globally-optimized distributed clustering method was introduced. The hierarchically-distributed peer-to-peer distributed clustering (HP2PC) architecture and algorithm were discussed. HP2PC aims to compute a single set of clusters across all nodes in the network, and addresses scalability and modularity through the concept of multiple overlays of node neighborhoods in the form of a hierarchy.

Distributed clustering is divided into two phases: at level 0 of the hierarchy, each neighborhood forms a set of cluster centroids through a distributed variant of the k-means algorithm. The centroids produced at level 0 neighborhoods are directly derived from the data at each node. At higher levels of the hierarchy, supernodes of lower level neighborhoods from higher level neighborhoods, at which the centroids at lower levels are meta-clustered. This process is repeated recursively until one set of centroids is computed at the root of the hierarchy, which reflects the global clustering of data in the whole network.

An analysis of the relationship between the number of nodes, the height of the hierarchy, and the network partitioning factor was given. Also computation and communication complexity were discussed.

Experimental results in chapter 6, section 6.4, illustrate that we can achieve good speedup with a large number of nodes, showing the scalability of the algorithm. Results also show a tradeoff between hierarchy height and quality of global clustering; taller hierarchies cause clustering quality to slightly drop, but achieve greater speedup.

CHAPTER 6

Experimental Results

Evaluation of the algorithms presented in this thesis is done through conducting a set of experiments using various document data sets. In this chapter, evidence in support of the three major contributions is presented, namely: keyphrase extraction from document clusters, collaborative peer-to-peer clustering, and hierarchically-distributed peer-to-peer clustering.

The accuracy of keyphrase extraction is important both for properly labeling document clusters as a post-processing step, which provides interpretation of clustering results to humans, and for providing the necessary cluster summarization functionality used in the collaborative peer-to-peer clustering method to share clustering results between nodes in a succinct form. Evaluation of CorePhrase shows significant keyphrase extraction accuracy in terms of *overlap* between the extracted keyphrases and the reference cluster topic, as well as in terms of *ranking* the most relevant phrases higher than non-relevant ones. Accuracy of CorePhrase is around double the that of keyword-centroid labeling method on average.

Collaborative clustering results show that collaboration between peers results in a

statistically-significant improvement of clustering quality compared to the initial clustering quality, reaching 15% improvement in F-measure quality. Statistical significance testing is performed to back those results. The effect of data distribution ratio is also tested by introducing overlap in data between nodes, which shows that as more data overlap exists, collaboration produces less improvement in clustering quality.

Finally, results on the hierarchically-distributed peer-to-peer clustering show that we can achieve good speedup with large number of nodes, reaching a speedup of 165 (over centralized version) for a 250-node network with a height of 5, showing the scalability of the algorithm. Results also show a tradeoff between hierarchy height and quality of global clustering; taller hierarchies cause clustering quality to slightly drop, but achieve greater speedup.

The rest of the chapter gives detailed description of the experiments, and provides interpretation of results and discussion of their implications.

6.1 Data sets

Experiments were performed on a number of document data sets with various characteristics and sizes. Table 6.1 lists the data sets used for evaluation. `YAHOO`, `20NG`, and `RCV1` are standard text mining data sets, while `CAN` and `UW` were manually collected and labeled; `SN` was manually collected but was already labeled. Below is is a brief description of each data set.

`Canada`

`CAN` is a collection of 151 web documents retrieved through the Google search engine by submitting six different queries about Canada (*e.g.* winter, snowboarding, river rafting,

Table 6.1: Data Sets

Data Set	Name	Type	Documents	Terms	Categories	Avgerage terms/doc
CAN	Canada	HTML	151	8,778	6	506
UW	UW	HTML	161	5,710	4	432
YAHOO	Yahoo! News	HTML	2,340	28,298	20	289
SN	SchoolNet	Metadata	2,371	7,166	17	145
20NG	20 newsgroups	USENET	18,828	91,652	20	151
RCV1	Reuters RCV1	Plain Text	23,149	47,236	103	196

etc.), and collecting the top 20-30 results. The topic of each class represents the terms used in the query. This data set provide realistic evaluation in scenarios involving online web mining applications (*e.g.* [89, 91]). It contains both related (*e.g.* snowboarding & winter) and unrelated categories (*e.g.* transportation & black bear attacks).

UW

UW is a collection of 161 web documents collected from intranet web sites at the University of Waterloo, with topics about various university services. This data set has moderate degree of overlap between the different categories, and was used in related work [40, 43]. The nature of this data set serves in the evaluation in scenarios involving data mining within an intranet information system. Both the UW and CAN data sets are available at http://pami.uwaterloo.ca/~hammouda/webdata/.

Yahoo! News

YAHOO is a collection of 2340 news articles from *Yahoo! News*. It contains 20 categories (such as health, entertainment, *etc.*), which have rather unbalanced distribution. It has been used in document clustering-related research, including [10, 12, 11, 96]. The data set

is available at
`ftp://ftp.cs.umn.edu/dept/users/boley/`.

`SchoolNet`

`SN` is a collection of 3271 metadata records collected from the SchoolNet learning resources web site (`http://www.schoolnet.ca/`). We extracted the fields containing text from the metadata records (title, description, and keywords) and combined them to form one document per metadata record. We used the 17 top-level categories from the SchoolNet data set.

`20 newsgroups`

`20NG` is the standard 20-newsgroup data set, which contains 18,828 documents from 20 Usenet newsgroups divided into 20 balanced categories. This data set is available at `http://people.csail.mit.edu/jrennie/20Newsgroups/`.

`Reuters RCV1`

`RCV1` is a subset of 23,149 documents selected from the standard Reuters RCV1 text categorization dataset, converted from the original Reuters RCV1 dataset by Lewis *et al* [75]. The documents in the RCV1 dataset are assigned multiple labels. In order to properly evaluate the clustering algorithms using single-label validity measures, we restricted the labels of the documents to the first document label that appears in the dataset.

Text Pre-processing

All text was preprocessed in the following way. First words were tokenized using a specially-built finite-state-machine tokenizer that can detect both alphanumeric and special entities

Table 6.2: Description of data and features used for keyphrase extraction

class	docs.	average words/doc.	candidate phrases	feature averages			
				df	w	pf	d
CAN							
canada transportation	22	893	24403	0.0934	0.2608	0.0007	0.4907
winter weather canada	23	636	5676	0.0938	0.2530	0.0019	0.5903
snowboarding skiing	24	482	990	0.0960	0.3067	0.0033	0.5215
river fishing	23	443	485	0.1129	0.2965	0.0042	0.5763
river rafting	29	278	599	0.0931	0.3680	0.0057	0.5612
black bear attacks	30	620	1590	0.0859	0.3593	0.0023	0.6024
UW							
Co-operative Education	54	251	1379	0.0511	0.2672	0.0082	0.5693
Career Services	52	508	4245	0.0473	0.2565	0.0031	0.5000
Health Services	23	329	351	0.1454	0.2707	0.0057	0.5170
Campus Network	32	510	14200	0.0810	0.2569	0.0020	0.5198

(such as currencies, dates, *etc.*). Then tokens were lower-cased, stop-words were removed, and finally the remaining words were stemmed using the popular Porter stemmer algorithm [90].

6.2 CorePhrase Keyphrase Extraction Results

A challenge faced while evaluating the cluster summarization accuracy of CorePhrase was the lack of datasets manually labeled with keyphrases. For the manually collected datasets CAN and UW it was easy to label the categories with keyphrases from the queries used to collect the documents. Thus most of the quantitative evaluation of CorePhrase is based on those two datasets. Nevertheless, we also show keyphrase extraction results for the 20NG dataset for qualitative evaluation.

Evaluation of cluster summarization in the collaborative (section 6.3) and hierarchical peer-to-peer clustering (section 6.4) schemes is based on using the results of the central-

ized CorePhrase algorithm as a baseline for comparison against the results obtained using distributed clustering.

Table 6.2 provides more detail about the two data sets used for quantitative evaluation, in terms of the number of documents in each class, average number of words per document, the number of candidate phrases, and average values of each keyphrase feature.

The documents in each class were processed by the four variants of the CorePhrase algorithm. The extracted keyphrases were ranked in descending order according to their score, and the top L keyphrases were selected for output by the algorithm. The baseline for comparison of the CorePhrase algorithm is a standard keyword-based extraction algorithm. The keyword centroid-based algorithm finds the centroid vector of a cluster represented as a set of keyword weights, and ranks them extracting the top weighted keywords in the cluster. This method is representative of most cluster labeling/summarization methods.

We introduce next the keyphrase extraction evaluation measures, followed by experimental results showing direct comparison with the keyword-based algorithm, then experiments illustrating the effect of the number of top keyphrases extracted (L), as well as the effect of introducing noise into a pure cluster on the accuracy of extraction.

6.2.1 Evaluation Measures

In addition to qualitative evaluation of the extracted keyphrases, we used two other extrinsic evaluation measures that quantitatively assess how well the extracted keyphrases relate to the topic of the original class or cluster. The extracted keyphrases are compared against a manually labeled keyphrase that represent the class topic. The metrics used are *overlap* and *best rank*, which are described next.

Overlap measures the similarity between each extracted keyphrase and the predefined topic phrase of the cluster. The similarity is based on the number of terms shared between

the two phrases. The overlap between an extracted keyphrase p_i and the topic phrase p_t is defined as:

$$\text{overlap}(p_i, p_t) = \frac{\mid p_i \cap p_t \mid}{\mid p_i \cup p_t \mid} \tag{6.1}$$

Evaluating each extracted keyphrase alone might not give a good idea of how the whole set of top k phrases fit the topic. To evaluate the top k keyphrases as a set, we take the average overlap of the whole set. This measure is essentially telling us how well the top keyphrases, as a set, *fit* the reference topic.

Best Rank gives an indication of how high the single keyphrase that best fits the topic is ranked. The best keyphrase is defined as the first keyphrase, in the top k, that has maximum overlap with the reference topic. Thus, the best rank for the set of top k phrases ($\mathbf{p}^k$) with respect to the reference topic p_t is defined as:

$$\text{bestRank}(\mathbf{p}^k, p_t) = \text{overlap}(p_{\max}, p_t) \cdot \left[1 - \frac{\text{rank}(p_{\max}) - 1}{k}\right] \tag{6.2}$$

where $p_{\max} \in \mathbf{p}^k$ is the first phrase with maximum overlap in the top k phrases; and $\text{rank}(p_{\max})$ is its rank in the top k. In other words, precision tells us how *high* in the ranking the best phrase appears. For example, if we get a perfect overlap in the first rank, precision is maximum. The lower the best phrase comes in the ranking, the lower the precision.

6.2.2 CorePhrase Accuracy

Table 6.3 shows the results of keyphrase extraction by the CorePhrase algorithm variants for four of the classes (two classes from `CAN`, and two classes from `UW`)[1]. Subjectively, the keyphrases extracted by the variants of the CorePhrase[2] algorithm are quite similar to the reference topic. The phrases in the results are shown in stemmed form, with stop words removed. In a real system the output of the algorithm would have to be in the original unstemmed form for presentation to the end user.

Table 6.4 shows the extracted keyphrases for 10 of the classes in the `20NG` dataset (using the CorePhrase-1M variant). The number of candidate keyphrases for each class is listed under the newsgroup name. Again, the top keyphrases exhibit high similarity to the corresponding newsgroup main topic. We can notice, however, that some irrelevant keyphrases make it to the top due to some artifacts of this particular dataset; *e.g.* the phrase "writes in article" appears at the top in a few categories due to a standard format used by news readers to quote a previous post. Also notice the second phrase in the comp.graphics category, "tiff: philosophical significance of 42", which turned out to be the subject of a heated debate in one of the threads, thus pushing it to the top. Otherwise, most of the extracted keyphrases look quite relevant to the main topic of the corresponding category, given that the number of candidate keyphrases to choose from is in the range of tens (sometimes hundreds) of thousands.

Evaluation based on the quantitative measures, overlap and best rank, is given in Table 6.5, which is illustrated also in figure 6.1. For each of the four variants of the CorePhrase algorithm, in addition to the baseline keyword centroid algorithm, we report the overlap

[1] A complete list of the results for the 10 classes can be found at:
`http://pami.uwaterloo.ca/~hammouda/corephrase/results.html`

[2] Throughout this discussion the name CorePhrase will refer to both CorePhrase-1 and CorePhrase-2, while CorePhrase-M will refer to both CorePhrase-1M and CorePhrase-2M, unless otherwise specified.

Table 6.3: Keyphrase Extraction Results, $L = 10$ [CAN, UW]

	CorePhrase-1	CorePhrase-2	CorePhrase-1M	CorePhrase-2M
			canada transporation	
1	canada transport	canada transport	transport canada	canada transport
2	panel recommend	canada transport act	canada transport	transport canada
3	transport associ	transport act	road transport	transport act
4	transport associ canada	transport associ	transport issu	transport issu
5	associ canada	panel recommend	govern transport	recommend transport
6	canada transport act	unit state	surfac transport	transport polici canada transport
7	transport act	transport associ canada tac	public transport	canadian transport
8	road transport	associ canada tac	transport public	transport public
9	transport infrastructur	canada tac	transport infrastructur	public transport
10	transport associ canada tac	public privat sector	transport passeng	transport infrastructur
			black bear attacks	
1	black bear	black bear	bear bear	bear bear
2	bear attack	bear attack	bear black bear bear	bear black bear bear
3	black bear attack	black bear attack	black bear bear	bear black bear
4	grizzli bear	black bear countri	bear black bear	black bear bear
5	bear safeti	grizzli bear	bear black	bear black
6	bear black bear	grizzli bear attack	black bear	black bear
7	bear countri	bear safeti	bear attack bear	bear attack bear
8	nation park	nation park	bear grizzli bear	bear grizzli bear
9	black bear countri	bear countri	bear encount bear	bear encount bear
10	deal bear	bear black bear	bear bear safeti	bear bear safeti
			health services	
1	health servic univers waterloo	health servic univers waterloo	servic univers	counsel servic
2	servic univers waterloo	servic univers waterloo	univers waterloo	assist student supplementari health
3	univers waterloo	univers waterloo	servic univers waterloo	student univers
4	health servic	health servic	health servic univers	supplementari health
5	health servic univers	health servic univers	health servic univers waterloo	page maintain chri
6	servic univers	servic univers	health servic	chri strome
7	page maintain chri strome	supplementari health	waterloo health	servic univers
8	page maintain	health care	copyright univers waterloo	student supplementari health plan
9	page maintain chri	student supplementari	copyright univers	health import
10	maintain chri strome	student supplementari health	student univers	student health
			campus network	
1	campu network	campu network	network network	network network
2	uw campu network	uw campu network	network uw network	network level network
3	uw campu	uw campu	network level network	network uw network
4	roger watt	network connect	uw network	network subscrib network
5	roger watt ist	level network	network uw	level network level network
6	watt ist	high speed	network subscrib network	level network
7	ip address	uw resnet	network assign network	network level
8	ip network	connect uw	network uw campu network	network assign network
9	high speed	area campu network	network level	extern network level network level network
10	request registr	switch rout	level network level network	network level network rout

Table 6.4: Keyphrase Extraction Results, CorePhrase-1M, $L = 15$ [20NG]

	comp.graphics	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	sci.crypt	sci.space
rank	74354	81101	66965	129520	114469
1	graphic librari	id vs scsi	centri 610	clipper chip	vandal sky
2	tiff philosoph signific 42	hard drive	monitor kept 24 hour	tap code	gamma rai burster
3	rumour 3do	17 monitor	x86 680x0 compar	white hous announc q clipper chip	hst servic mission schedul 11 dai
4	24 bit	local bu	centri 650	netcom com	access digex
5	help need	cd rom	duo 230	kei escrow	1 billion year
6	3d graphic	video card	se 30	chip encrypt	space market
7	cview answer	id drive	lc iii	secret algorithm clipper chip crypto kei	edu write
8	polygon routin	western digit	cach card	clipper chip crypto kei	dc x
9	im look	diamond stealth	quadra scsi problem	secret algorithm clipper chip	space station
10	newsgroup split	date stuck	mac plu	clipper chip crypto kei escrow	prb access digex
11	comp graphic	hard disk	hard drive	crypto kei	digex net
12	recommend 3d graphic librari	ibm pc	non appl	clipper consid harm	nasa gov
13	polygon routin need	isa bu	mac ii	wiretap chip	space market wonderful
14	file format	write articl	new duo	keyseach machin	aurora alaska
15	fast polygon routin	soundblast irq port set	mac portabl	david sternlight	temporari orbit
	comp.sytalk.politics.mideast	rec.autos	rec.motorcycles	rec.sport.baseball	rec.sport.hockey
rank	232619	98341	85662	104535	149481
1	isra terror	warn read	shaft drive	jewish basebal player	nhl team
2	europ vs muslim bosnian	automot concept	shaft drive wheeli	basebal player	plu minu
3	muslim bosnian	v4 v6 v8 v12 vx	drive wheeli	jack morri	european nhl
4	write articl	write articl	write articl	brave updat	nhl team captain
5	israel expans	chang oil	counterst faq post	red sox	goali mask
6	u s	chang oil self	need advic	best homerun	plu minu stat
7	center polici research	manual shift	moa member	jai vs indian seri	edu write
8	edu write	rec auto	good reason wave	america team	team captain
9	igc apc org	question insur compani	faq post	hbp bb big cat	octopu detroit
10	freedom u s	opel owner	polic offic	bat 4th	don cherri
11	armenia sai shoot	radar detector	need advic ride pillion	game length	red wing
12	polici research cpr	ohio state	riceburn respect	barri bond bat 4th	stanlei cup
13	turkish plane	dirti diesel	bmw moa member	dave kingman	game plai
14	human right	1994 mustang	maxima chain wax	time best player	abc coverag
15	final solut	plymouth sundanc dodg shadow	dog attack	speed game	nhl letter

and best rank. The average overlap is taken over the top 10 keyphrases/keywords of each class, with the maximum overlap value (best phrase) also shown. Total averages per data set and overall average is also reported.

The first observation is that CorePhrase performs consistently better than the keyword centroid method. This is attributed to the keyphrases being in greater overlap with the reference topic than the naturally-shorter keywords. An interesting observation also is that CorePhrase-M, which is based on weighted words for phrase-scoring, and the keyword centroid follow the same trend. This is due to the link between the phrase scores and their constituent word scores.

The second observation is that the variants of the algorithm that use the depth feature (CorePhrase-2 and CorePhrase-2M) are consistently better than those that do not use the depth feature (CorePhrase-1 and CorePhrase-1M) in terms of both overlap and best rank. This is attributed to the fact that some common phrases usually appear at the end of each document (such as "last updated", "copyright", the name of the web site maintainer). If depth information is ignored, these phrases make their way up the rank (*e.g.* the phrase "roger watt" in **campus network** cluster, which is the name of the network maintainer that appears at the end of each document.) If depth information is taken into consideration, these phrases are penalized due to their appearance at the end of the document.

Another observation is that the four variants of the algorithm were able to discover the topic of the cluster and rank it in the top 10 keyphrases, which can be deduced from the maximum overlap value. CorePhrase is somewhat better than its word-weighted counterpart (CorePhrase-M) in extracting the best phrase and ranking it among the top 10, where it achieves 97% overlap on average for the best phrase. The word-weighted variant achieves 83% maximum overlap on average for the best phrase.

However, if we look at the set of the top 10 extracted phrases as a whole and not just

Table 6.5: Performance of the CorePhrase algorithm, ($L = 10$) [CAN, UW]

	CorePhrase-1		**CorePhrase-2**		**CorePhrase-1M**		**CorePhrase-2M**		**Keyword Centroid**	
class	overlap (avg,max)	bestRank	overlap (avg,max)	bestRank	overlap (avg,max)	bestRank	overlap (avg,max)	bestRank	overlap (avg,max)	bestRank
Dataset 1										
canada transportation	(0.45,1.00)	1.00	(0.32,1.00)	1.00	(0.47,1.00)	1.00	(0.57,1.00)	1.00	(0.22,0.50)	0.5
winter weather canada	(0.22,0.67)	0.60	(0.28,0.67)	0.60	(0.00,0.00)	0.00	(0.00,0.00)	0.00	(0.00,0.00)	0.0
snowboarding skiing	(0.37,1.00)	1.00	(0.47,1.00)	1.00	(0.58,1.00)	0.90	(0.58,1.00)	0.90	(0.24,0.50)	0.5
river fishing	(0.41,1.00)	0.90	(0.41,1.00)	0.80	(0.43,1.00)	0.60	(0.39,1.00)	0.60	(0.14,0.50)	0.5
river rafting	(0.38,1.00)	1.00	(0.42,1.00)	1.00	(0.68,0.67)	0.90	(0.65,0.67)	1.00	(0.32,0.50)	0.5
black bear attacks	(0.45,1.00)	0.80	(0.48,1.00)	0.80	(0.47,1.00)	0.60	(0.51,1.00)	0.60	(0.25,0.33)	0.33
data set 1 average	(0.38,0.95)	0.88	(0.39,0.95)	0.87	(0.44,0.78)	0.67	(0.45,0.78)	0.68	(0.20,0.39)	0.39
Dataset 2										
co-operative education	(0.38,1.00)	0.20	(0.47,1.00)	0.30	(0.55,1.00)	0.80	(0.83,1.00)	0.90	(0.41,0.50)	0.3
career services	(0.37,1.00)	0.70	(0.42,1.00)	0.70	(0.58,1.00)	0.90	(0.43,1.00)	0.90	(0.26,0.50)	0.5
health services	(0.28,1.00)	0.70	(0.38,1.00)	0.70	(0.32,1.00)	0.50	(0.21,0.33)	0.33	(0.10,0.50)	0.5
campus network	(0.23,1.00)	1.00	(0.40,1.00)	1.00	(0.38,0.67)	0.20	(0.33,0.50)	0.50	(0.12,0.50)	0.5
data set 1 average	(0.31,1.00)	0.65	(0.42,1.00)	0.68	(0.46,0.92)	0.60	(0.45,0.71)	0.66	(0.22,0.46)	0.45
overall average	**(0.35,0.97)**	**0.79**	**(0.40,0.97)**	**0.79**	**(0.45,0.83)**	**0.64**	**(0.45,0.75)**	**0.67**	**(0.21,0.42)**	**0.41**

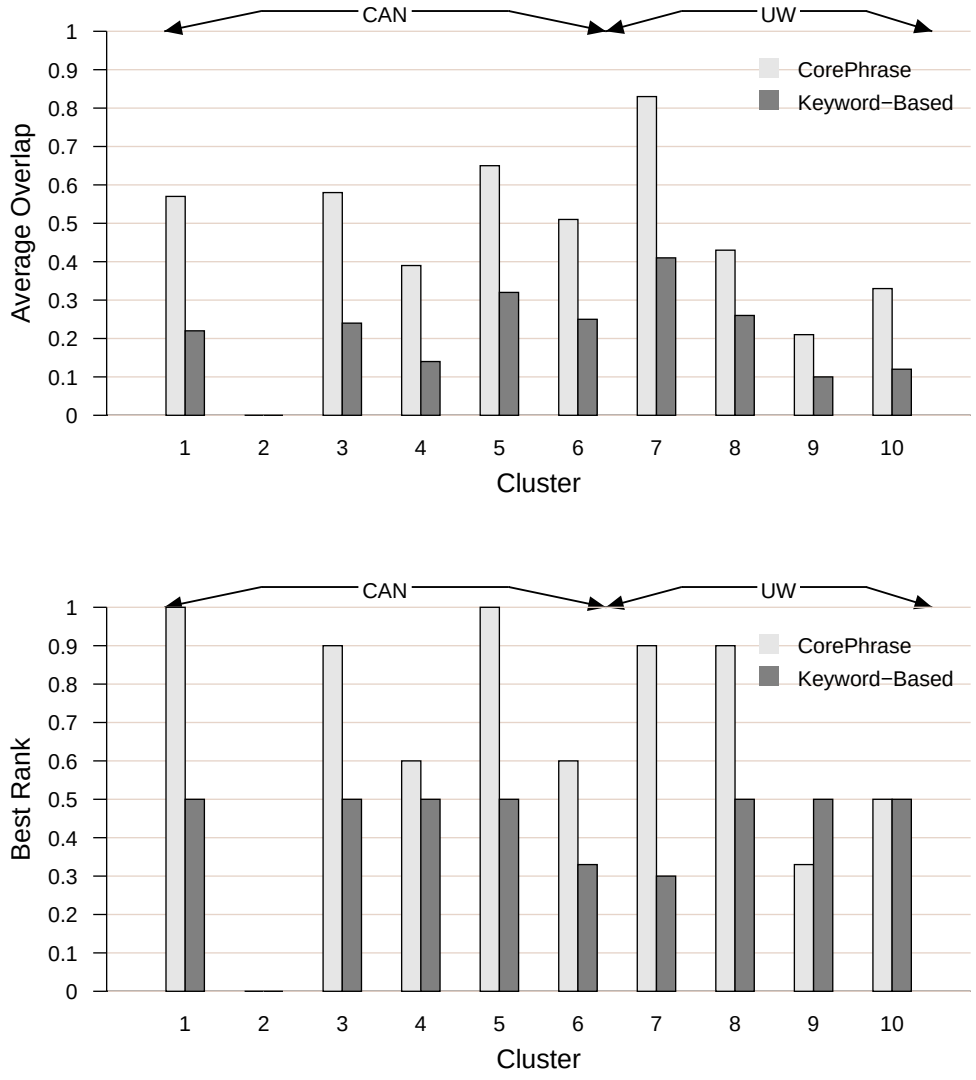


Figure 6.1: Keyphrase Extraction Accuracy Comparison, Top 10 Keyphrases [CAN, UW]

the best phrase, the word-weighted variant achieves better performance in terms of average overlap (45% for CorePhrase-M against 40% for CorePhrase). This is attributed to the fact that keyphrases extracted by the word-weighted version will always contain heavily weighted words, which often overlap with the reference topic. This means that CorePhrase-M will consistently extract phrases containing words found in the reference topic, but which do not necessarily constitute the best descriptive keyphrases. This drawback manifests itself when there are few words which occur very frequently throughout the candidate phrases, but are not part of the reference topic. In this case the algorithm will rank up irrelevant phrases which contain those words due to their heavy weight. (An example is the **winter weather canada** cluster.)

A final observation is that CorePhrase consistently achieves better best rank than CorePhrase-M (79% for CorePhrase against 67% for CorePhrase-M.) This means that CorePhrase does not only find the best keyphrase, but ranks it higher than CorePhrase-M.

To summarize these findings: (a) CorePhrase is more accurate than keyword-based algorithms; (b) using phrase depth information achieves better performance; (c) using word-weights to rank phrases usually produces a better *set* of top phrases; however, ignoring the word-weights usually produces the best descriptive phrase and ranks it higher; and (d) in most cases, CorePhrase is able to identify the reference topic in the top few keyphrases.

6.2.3 Number of Extracted Keyphrases

So far we evaluated the accuracy of CorePhrase with respect to the relevance of the extracted keyphrases to the cluster topic using a fixed number of top keyphrases, usually 10 or 15 keyphrases. In order to test if we could have used a larger number to get more relevant keyphrases, we report the trend of the average overlap measure against the number

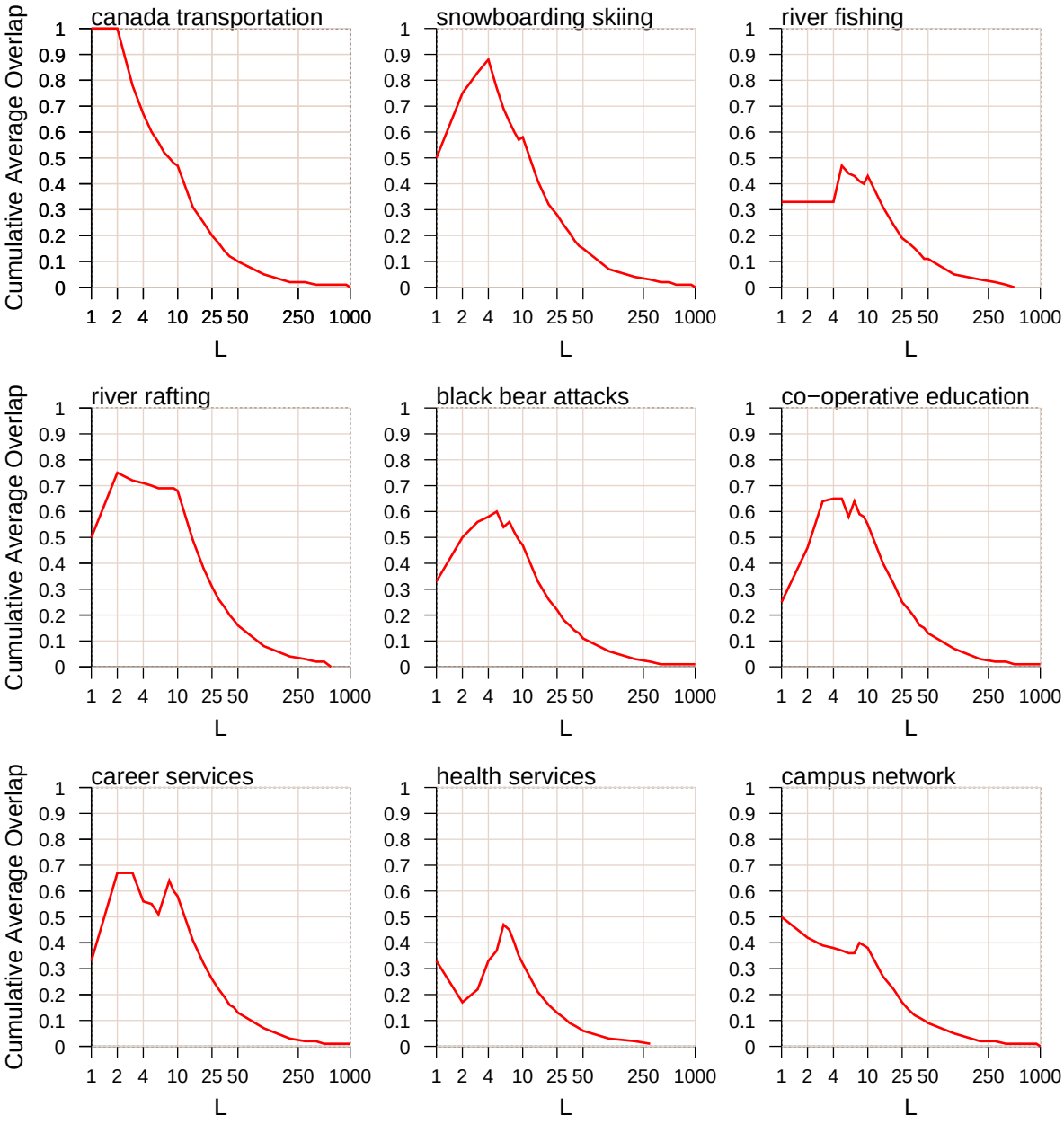


Figure 6.2: CorePhrase Accuracy vs. Number of Top Keyphrases [CAN, UW]

of extracted keyphrases, L. Figure 6.2 illustrates this trend using 9 classes from `CAN` and `UW`. The number of extracted keyphrases, L, is reported on the horizontal axis (log scale), while the cumulative average overlap is reported on the vertical axis.

The observation made here is that the top several keyphrases are responsible for the peak of the average overlap, before the trend goes down quickly; *i.e.* most of the relevant keyphrases are correctly ranked at the top, while irrelevant phrases are pushed down the ranking. This suggests that it is reasonably safe to assume that the top 10, 15, or 20 keyphrases are usually sufficient to properly summarize a cluster of documents.

6.2.4 Effect of Cluster Impurity

It is important to investigate how well CorePhrase behaves in the presence of noise. In previous experiments we only introduced pure clusters to CorePhrase; *i.e.* documents from the same class. A set of experiments were performed where we introduced impurity into a cluster and the accuracy of extracted keyphrases is measured. In such experiments, accuracy is measured as the percentage of extracted keyphrases that are common with those extracted from a pure cluster; *i.e.*

$$\% \text{ of correct keyphrases} = \frac{|\text{keyphrases common with pure cluster}|}{L}$$

In those experiments, we started with a pure cluster and incrementally replaced documents from the pure cluster with documents from other classes. Two document replacement scenarios were tested: (1) random replacement with documents from a single class, and (2) random replacement with documents from multiple classes.

Figure 6.3 shows the results of those experiments. We can observe that in both replacement scenarios, the impurity introduced into the pure cluster affects the accuracy of

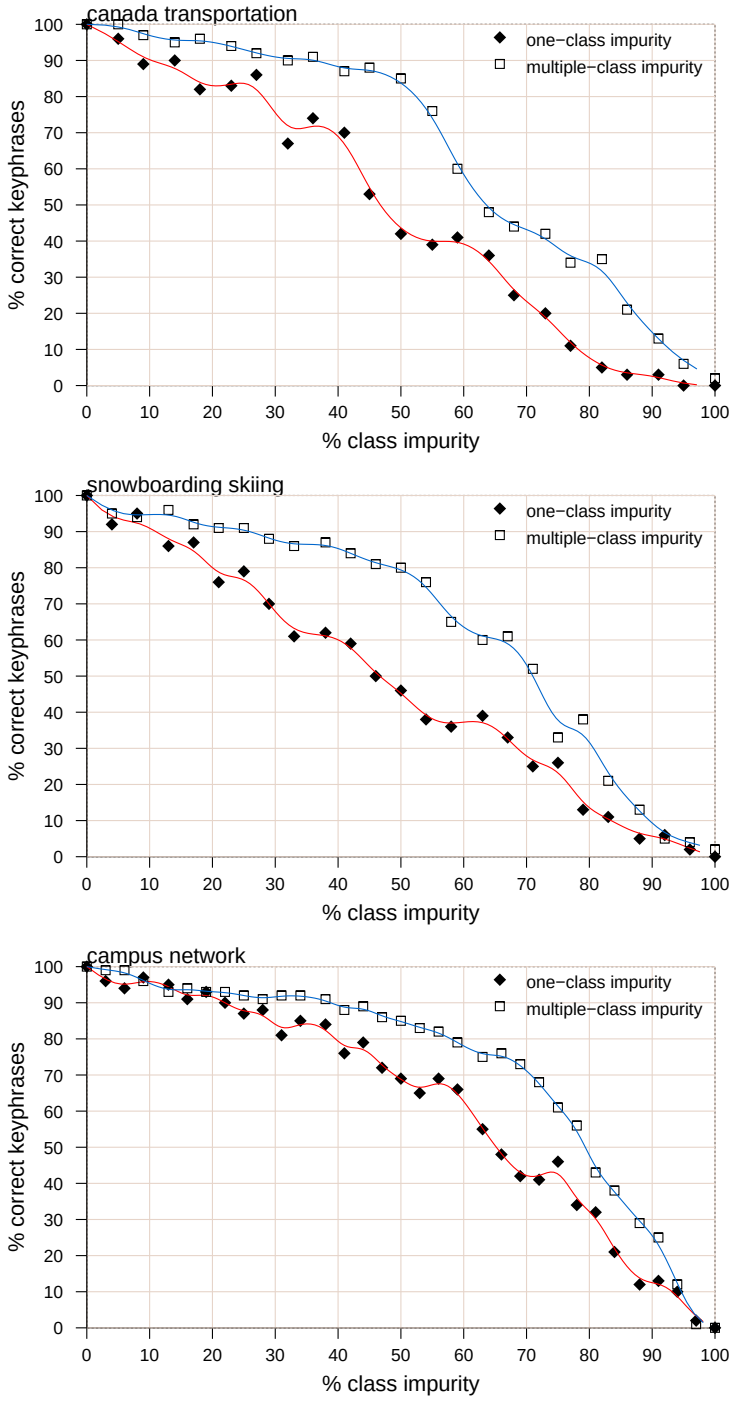


Figure 6.3: CorePhrase Accuracy vs. Class Impurity [CAN, UW]

extracted keyphrases, which is expected due to the gradually disappearing support of the correct keyphrases.

The second observation is that the accuracy of CorePhrase seems to tolerate noise introduced from multiple classes as opposed to noise introduced from a single class. When introducing single-class noise, keyphrases from the noisy class tend to compete with the original class. Introducing multiple-class noise, on the other hand, does not provide enough support for competing keyphrases to appear quickly, since the introduced documents do not share much in common.

Thus, unless noise is consistently introduced from a single class of documents, CorePhrase tolerates random impurity in clusters by suppressing the usually low-support phrases in such noisy documents.

6.2.5 CorePhrase Scalability

To test the scalability of CorePhrase, we applied the algorithm with 3 of the larger datasets: `YAHOO`, `SN`, and `20NG`[3]. Usually CorePhrase works on one class or cluster at a time, but for the sake of scalability demonstration, we combined all classes from each dataset into one set which was processed by the algorithm. We also tested the performance against `CAN` and `UW`, but the results were not included because the number of documents in those datasets is relatively small, and usually takes a few seconds to complete.

Figure 6.4 shows the time performance of CorePhrase against the three datasets. The results were obtained using an Intel Core 2 Duo computer, with 4 GB of RAM. The first observation is that CorePhrase scales well with the number of documents in all datasets. It can process several hundred documents in under a minute; or several thousands of

[3]The `RCV1` dataset was not used in this experiment because it was in a format where the original order of words in documents was not preserved.

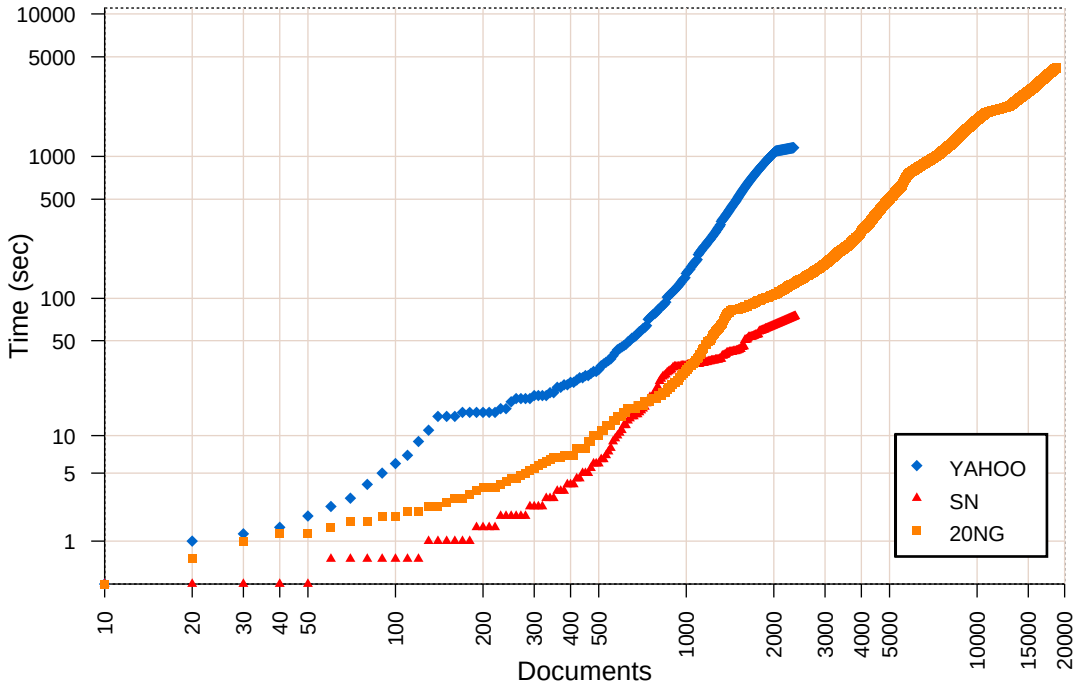


Figure 6.4: CorePhrase Time Performance [YAHOO, SN, 20NG]

documents in a few minutes.

The second observation is the difference between datasets. CorePhrase takes less time to process datasets with fewer words per document than those with more words. While YAHOO has almost twice the number of words per document as SN, it takes about 15 times the time to process. SN and 20NG are roughly close to each other in performance (they also have similar average words per document). Thus, while CorePhrase can scale well with large number of documents, it may need improvement with respect to the dimensionality of the documents, which we leave for future work.

In the next two sections we evaluate both the collaborative peer-to-peer and hierarchical

peer-to-peer clustering algorithms, and examine how CorePhrase can be used in distributed clustering environments for both improving clustering accuracy and producing meaningful summaries for distributed clusters.

6.3 Collaborative P2P Clustering Results

In this section we introduce the experiments performed to evaluate the collaborative P2P document clustering algorithm. We tested three network configurations and measured the difference between the initial and final clustering quality to see the effect of collaboration.

6.3.1 Results and Discussion

Table 6.6 shows the results obtained for three network configurations: 3-Node, 5-Node, and 7-Node for `YAHOO`, while Table 6.7 shows similar results for `20NG`. The results reported were obtained using a random data distribution ratio $\alpha = 1/N_P$; *i.e.* the data was equally and randomly partitioned over all nodes.

For each configuration five runs were performed, and the average was taken to alleviate any bias that may result from random data partitioning and the insertion order of the documents to the SHC clustering algorithm. The results of the average initial and final (after aggregating peer recommendations) F-measure (F) and Entropy (E) per node are reported, along with overall averages over the nodes. The difference between the final and initial measure values is reported as the improvement in accuracy. The F-measure and Entropy were introduced in chapter 2.

Figure 6.5 illustrates the mean initial and final clustering F-measure values for each of the network configurations, bounded by their standard deviation (error bars). Results for both `YAHOO` and `20NG` are shown. Figures 6.6 and 6.7 illustrate the same results (F-measure

Table 6.6: Collaborative P2P Clustering Results [YAHOO]

	3-Node				5-Node				7-Node			
	Initial		Final		Initial		Final		Initial		Final	
Node	F	E	F	E	F	E	F	E	F	E	F	E
1	0.59	0.20	0.65	0.18	0.44	0.25	0.58	0.20	0.26	0.36	0.40	0.26
2	0.61	0.19	0.66	0.18	0.47	0.23	0.59	0.19	0.29	0.32	0.42	0.21
3	0.59	0.18	0.64	0.14	0.42	0.28	0.53	0.22	0.30	0.31	0.48	0.19
4					0.42	0.29	0.54	0.21	0.28	0.34	0.43	0.22
5					0.43	0.28	0.55	0.19	0.27	0.35	0.43	0.24
6									0.28	0.34	0.42	0.23
7									0.32	0.29	0.49	0.17
Average	0.60	0.19	0.65	0.17	0.44	0.27	0.56	0.20	0.29	0.33	0.44	0.22
Std Dev	±0.01	±0.01	±0.01	±0.02	±0.03	±0.02	±0.03	±0.01	±0.03	±0.03	±0.05	±0.03
Improvement			+5%	-2%			+12%	-6%			+15%	-11%

Table 6.7: Collaborative P2P Clustering Results [20NG]

	3-Node				5-Node				7-Node			
	Initial		Final		Initial		Final		Initial		Final	
Node	F	E	F	E	F	E	F	E	F	E	F	E
1	0.44	0.27	0.47	0.25	0.38	0.29	0.44	0.23	0.31	0.32	0.45	0.25
2	0.45	0.30	0.48	0.25	0.40	0.32	0.49	0.26	0.43	0.33	0.48	0.26
3	0.43	0.28	0.46	0.24	0.34	0.32	0.45	0.26	0.35	0.34	0.47	0.23
4					0.46	0.27	0.49	0.28	0.36	0.31	0.47	0.24
5					0.40	0.32	0.46	0.27	0.34	0.31	0.44	0.21
6									0.38	0.28	0.47	0.22
7									0.26	0.32	0.44	0.30
Average	0.44	0.28	0.47	0.25	0.40	0.30	0.47	0.26	0.35	0.32	0.46	0.25
Std Dev	±0.03	±0.03	±0.04	±0.03	±0.06	±0.05	±0.07	±0.04	±0.08	±0.05	±0.08	±0.07
Improvement			+3%	-3%			+7%	-4%			+11%	-7%

only), showing averages of the initial F-measure (solid bars) and final F-measure (difference bars) of all three network configurations on the same graph for comparison.

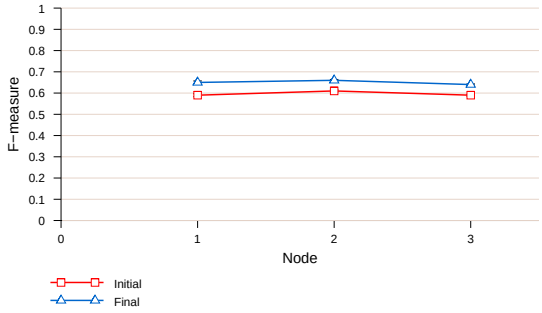
It should be noted that the evaluation of the final clustering after aggregation of peer-positive documents takes into consideration that the initial document class distribution at the local node has been changed, and updates the final number of documents in each class. This is essential for the initial and final F-measure and Entropy calculations to be correct.

The results are interesting, and show an improvement in both evaluation measures per node and on average. A very important observation is that networks with fewer nodes appear to have higher absolute accuracy (both initial and final) compared to networks with

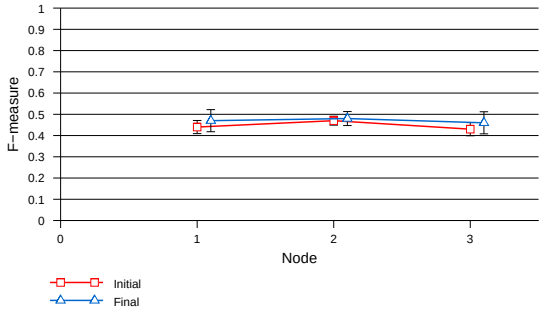
larger number of nodes; *e.g.* the 3-Node network has initial F of 59% and final of 65%, as opposed to 30% and 45% in the 7-Node network. However, networks with larger number of nodes exhibit greater improvement in the final clustering compared to the initial clustering, as can be seen from the improvement percentages; *e.g.* in the case of the YAHOO dataset the 7-Node network has an improvement of 15% in F compared to just 6% improvement in the 3-Node network.

This observation is attributed to the fact that, in networks with fewer nodes, each node has access to larger percentage of the data than those in larger networks. This results in smaller networks being able to build better clusters. The ideal case is when there is only one node, in which case the one and only node has complete knowledge about all the data and thus can build the best clusters. However, the real benefit of the system is observed in situations involving larger networks (typical case). Each node in larger networks has limited view of the global data collection, but through distributed clustering they are able to have better visibility of global data through peer recommendation.

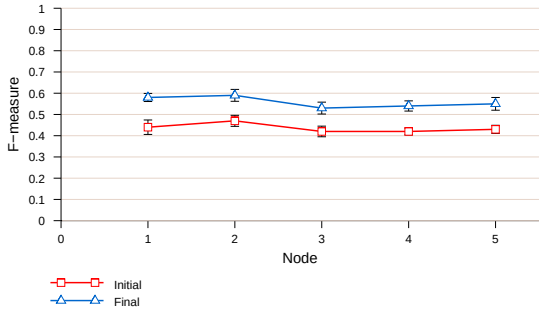
The results, however, show a difference between the two data sets under investigation. For the YAHOO dataset the improvement in clustering quality is more noticeable than in the case of the 20NG data set. This reflects some data set characteristics. The 20NG data set has much larger number of documents, with almost even distribution of documents among classes. On the contrary, the YAHOO dataset has fewer documents with unbalanced class distribution. Most clusters created in the case of the YAHOO set encompassed much of the smaller classes resulting in higher evaluation measure. On the other hand, in the 20NG case classes were more fragmented among clusters, causing lower clustering quality. In both cases, however, improvement in clustering quality is noticeable, albeit a smaller change in the case of the 20NG data set.



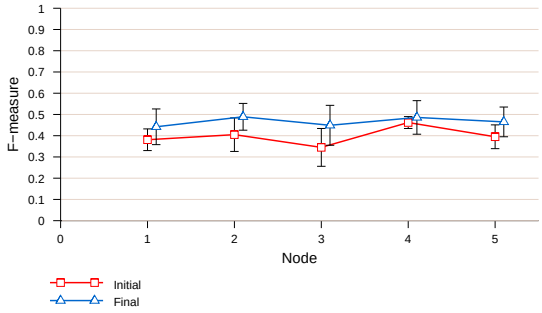
(a) 3 nodes, YAHOO



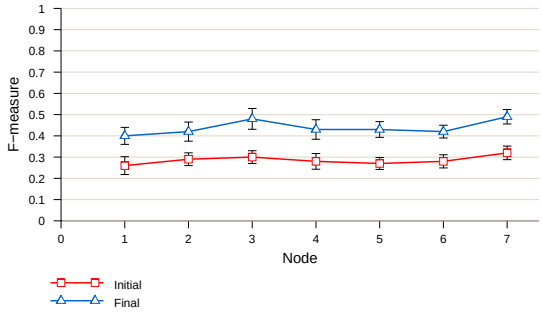
(b) 3 nodes, 20NG



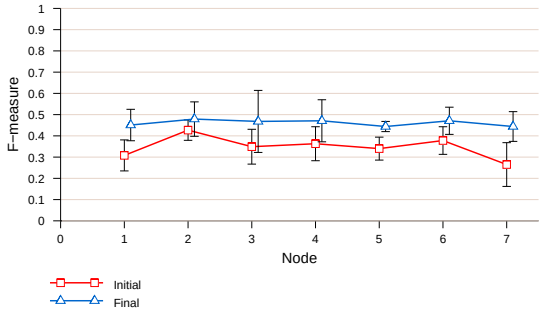
(c) 5 nodes, YAHOO



(d) 5 nodes, 20NG



(e) 7 nodes, YAHOO



(f) 7 nodes, 20NG

Figure 6.5: Average F-measure (Initial and Final). (a), (c), and (e) YAHOO; (b), (d), and (f) 20NG.

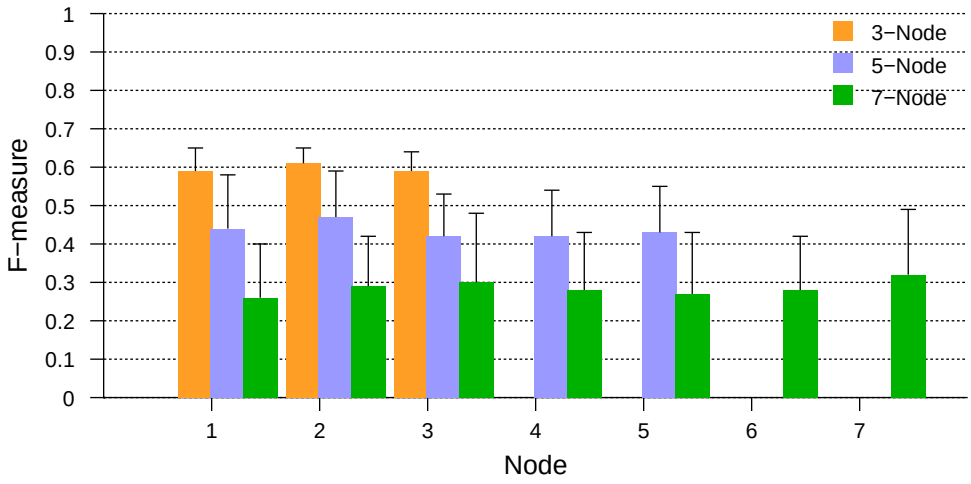


Figure 6.6: Clustering Accuracy Improvement – F-measure [YAHOO]

6.3.2 Significance Testing

To back the claim of clustering quality improvement, statistical significance testing is presented here, where the average final F-measure is compared to the average initial F-measure. This is a comparison of two means test that enables us to calculate the confidence intervals for the difference between the two means.

Our null hypothesis (which we will argue to be rejected in favor of the alternate hypothesis) is that the average F-measure for the initial clustering and the final clustering are the same; i.e.

$$H_0 : \bar{F}_f = \bar{F}_i \tag{6.3}$$

where $\bar{F}_f$ is the average F-measure for the final clustering over all runs for all nodes, and $\bar{F}_i$ is the corresponding average initial F-measure. The alternate hypothesis will be

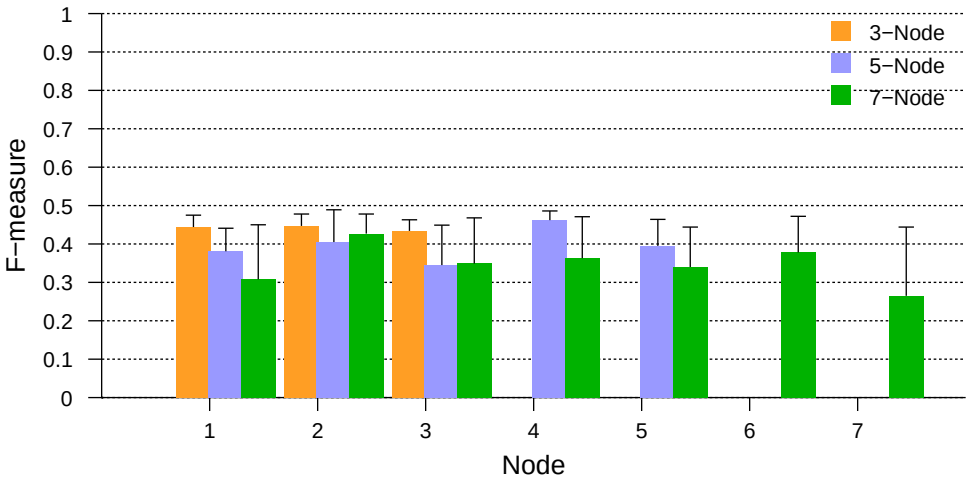


Figure 6.7: Clustering Accuracy Improvement – F-measure [20NG]

that the average final F-measure is greater than the initial average F-measure; i.e.

$$H_a : \bar{F}_f > \bar{F}_i \tag{6.4}$$

Since the actual underlying means and standard deviations are not known, we are going to use a **two-sample t statistic** (instead of a two-sample z statistic), in which the population standard deviation is estimated by the calculated standard deviations s_1 and s_2 from the samples. The t statistic is given by:

$$t = \frac{(\bar{x_1} - \bar{x_2}) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{6.5}$$

where $\bar{x_1}$ and $\bar{x_2}$ are the calculated means of the two populations, μ_1 and μ_2 are the actual means, s_1 and s_2 are the calculated standard deviations, and n_1 and n_2 are the

sample sizes from the two populations.

For the 3-node network case there are 15 sample pairs (of initial and final F-measure): 3 nodes x 5 runs. The average initial F-measure is $\bar{F}_i = 0.60$, while the average final F-measure is $\bar{F}_f = 0.65$. The calculated standard deviations are $s_i = 0.0086$ and $s_f = 0.0098$ for the initial and final measures, respectively. The t statistic for the 3-node case under the null hypothesis ($\mu_1 - \mu_2 = 0$) is thus

$$t = \frac{(\bar{F}_f - \bar{F}_i) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_i^2}{n_i} + \frac{s_f^2}{n_f}}} = \frac{(0.65 - 0.60) - 0}{\sqrt{\frac{0.000074}{15} + \frac{0.000099}{15}}}$$
$$= 14.18$$

Using the $t(14)$ distribution, we see that $2 \cdot P(t \geq 14.18)$ is < 0.001, which allows us to reject the null hypothesis in favor of the alternate hypothesis, indicating a significant difference between the two means. Thus the difference between the initial and final F-measures is statistically significant.

The confidence interval of the difference between the two means at a confidence level C (usually 95%) is given by

$$(\bar{x_1} - \bar{x_2}) \pm t^* \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} \tag{6.6}$$

where t^* is the upper $(1 - C)/2$ critical for the t distribution with k degrees of freedom (number of samples - 1). To compute a 95% confidence interval for the difference between the two means, we first note that the 0.025 ($(1 - C)/2$) critical value t^* of the $t(14)$ distribution is 2.14479 (from standard t distribution tables). This gives us the interval $(0.041, 0.0556)$; i.e. the difference between the initial and final average F-measure is $0.048 \pm$

Table 6.8: Significance Tests and Confidence Interval [YAHOO]

Network	$\bar{F}_i$	$\bar{F}_f$	$\bar{\Delta F}$	s_i	s_f	t	P-value	error	ΔF confidence interval
3-Node	0.603	0.651	0.048	0.0086	0.01	14.18	< 0.001	0.0073	(0.0410, 0.0556)
5-Node	0.437	0.557	0.120	0.0292	0.0342	13.33	< 0.001	0.0240	(0.0959, 0.1438)
7-Node	0.287	0.440	0.153	0.0324	0.0486	15.46	< 0.001	0.0308	(0.1218, 0.1834)

Table 6.9: Significance Tests and Confidence Intervals [20NG]

Network	$\bar{F}_i$	$\bar{F}_f$	ΔF	s_i	s_f	t	P-value	error	ΔF confidence interval
3-Node	0.441	0.472	0.030	0.0266	0.0435	2.300	0.019	0.0282	(0.0020, 0.0585)
5-Node	0.400	0.466	0.068	0.0703	0.0746	3.327	0.001	0.0546	(0.0136, 0.1228)
7-Node	0.347	0.459	0.112	0.0831	0.0798	5.737	< 0.001	0.0607	(0.0510, 0.1724)

0.0073.

The same test can be applied to the two other network configurations of 5 nodes and 7 nodes. The calculations are summarized in table 6.8. An interesting observation is that as the average improvement in clustering quality increases with the increase in the number of nodes, the confidence interval of the improvement also increases, indicating a larger margin of error (although still acceptable). We can attribute this to the variation in the output of the experiments when using larger number of nodes, since they tend to each have less exposure to the data than with the networks with fewer nodes, resulting in larger variation on judgement of a proper clustering solution.

The same analysis has been applied to the 20NG dataset, and the results are reported in Table 6.9. The tests show that the improvement is still statistically significant, at least at the 0.05 level for the 3-node case, and at the 0.01 level for the 5-node and 7-node cases.

6.3.3 Effect of Data Distribution Ratio

A set of experiments have been conducted to test the effect of increasing the data distribution ratio among nodes, α. Figure 6.8 illustrates the quality trend against this ratio for the YAHOO dataset. The line graphs in the figure represent the final F-measure of each network configuration for various distribution ratios. The vertical error bars show the difference between the initial and final clustering F-measure quality.

As noted earlier, the figure confirms that networks with fewer nodes (*e.g.* 3-Node) maintain higher absolute accuracy for the range of distribution ratios. It also confirms our observation that networks with more nodes exhibit greater relative increase in clustering quality after aggregating peer recommendations. It is also interesting to note that as the distribution ratio reaches its maximum (all nodes having access to 100% of the data), all networks tend to have the same (maximum) performance, which is logical since at this maximum distribution ratio ($\alpha = 1$) each node has access to the full data set.

Notice also that at the same distribution ratio, networks with higher number of nodes result in higher improvements in the final clustering result quality than those with fewer number of nodes. This is attributed to the larger coverage of data found in networks with higher number of nodes.

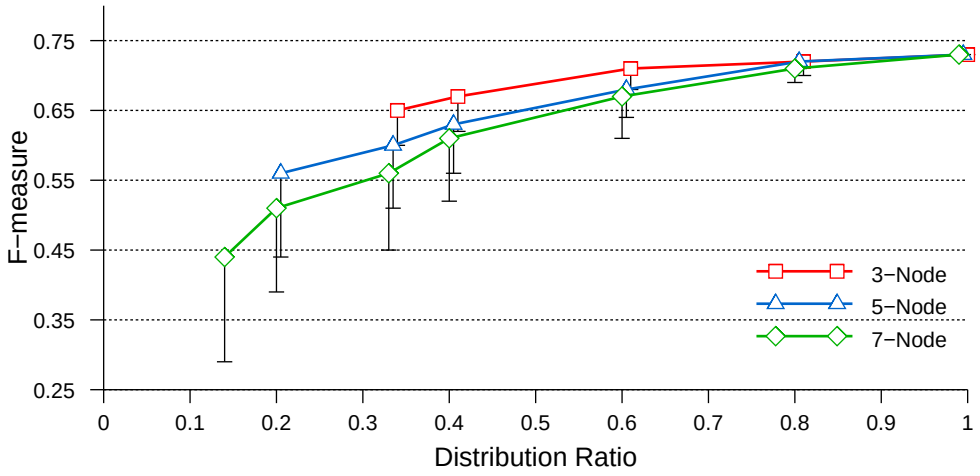


Figure 6.8: Effect of Data Distribution Ratio [YAHOO]

6.4 Hierarchically-Distributed P2P Clustering Results

6.4.1 Experimental Setup

A simulation environment was used for evaluating the HP2PC algorithm. During simulation, data was partitioned randomly over all nodes of the network. The number of clusters was specified to the algorithm such that it corresponds to the actual number of classes in each dataset. A random set of centroids were chosen by each supernode and the centroids were distributed to all nodes in its neighborhood at the beginning of the process. Clustering was invoked at level 0 neighborhoods, and was propagated to the root of the hierarchy as described in section 5.3.

6.4.2 Evaluation Measures

Three aspects of the algorithm were evaluated: clustering accuracy, speedup, and distributed summarization accuracy. For evaluating clustering accuracy we relied on both external (F-measure) and internal evaluation measures (Separation Index). Detailed discussion of clustering evaluation measures is given in section 2.1.4.

We evaluated the quality of clustering at different levels of the hierarchy. At level $h = 0$, we evaluated the quality of clustering for each neighborhood, with respect to the subset of the data in the neighborhood; *i.e.*

$$F_Q = F(C_Q, D_Q)$$

where C_Q is the set of clusters obtained for neighborhood Q, and D_Q is the union of data sets of all nodes in that neighborhood ($D_Q = \bigcup_{i \in Q} D_i$).

At level $h > 0$, we evaluated the the clustering acquired by a supernode with respect to the data subset of the nodes at the level 0 reachable from the supernode. Thus, evaluation of the clustering acquired at the root node reflects the quality with respect to the whole data set.

Speedup is a measure of the relative increase in speed of one algorithm over the other. For evaluating HP2PC, it is calculated as the ratio of time taken in the centralized case (T_c) to the time taken in the distributed case (T_d), including communication time; *i.e.*

$$S = \frac{T_c}{T_d}. \tag{6.7}$$

To take communication time into consideration in the simulations, we factored the time

taken to transmit a message from one node to another on a 100 Mbps link[4]. Thus, the time required to transmit a message of size $|M|$ bytes is calculated as:

$$T_M = |M|/(100,000,000/8) \text{ seconds.}$$

During simulation, each time a message is sent from (or received by) one node to another, its time is calculated and is added to the total time taken by that node. Since in a real environment all nodes on the same level of the hierarchy run in parallel, the total time taken by that level is calculated as the maximum time taken by any node on the same level. Time taken by different levels is added to arrive at the global T_d.

For cluster summarization accuracy, evaluation of the produced cluster summaries was based on how much the extracted keyphrases agree with the centralized version of CorePhrase when run on the centralized cluster. Assume HP2PC produced a cluster c_k that spanned N_p nodes, each holding subset of the documents, D_{ki}, from that cluster. If all documents were pooled into a centralized cluster, we have D_k documents in that cluster. The percentage of correct keyphrases is calculated as:

$$\% \text{ correct keyphrases} = \frac{\text{CorePhrase}(D_k) \cap \text{DistCorePhrase}(\{D_{ki}\})}{L}$$

where L is the maximum number of top keyphrases extracted.

In the next subsections, we evaluate the effect of network size on clustering accuracy, the effect of scaling the hierarchy height, the quality of clustering at different levels within a single hierarchy, and the accuracy of distributed cluster summarization using the distributed CorePhrase algorithm.

[4]This is a simplified assumption. Real networks exhibit communication overhead due to network protocols and network congestion.

Table 6.10: Accuracy and Performance of HP2PC [YAHOO]

	F-measure			SI			Speedup		
Nodes	H=1	H=2	H=3	H=1	H=2	H=3	H=1	H=2	H=3
1	0.683	0.679	0.705	0.035	0.035	0.035	1.00	1.00	1.00
5	0.706	0.655	0.643	0.052	0.051	0.184	3.11	3.80	4.02
10	0.685	0.641	0.627	0.036	0.104	0.171	4.67	6.90	7.16
15	0.681	0.634	0.579	0.021	0.122	0.236	7.15	10.50	11.87
20	0.667	0.612	0.541	0.022	0.168	0.233	12.68	15.10	16.48
25	0.711	0.593	0.535	0.031	0.190	0.248	15.33	18.30	20.10
30	0.693	0.589	0.531	0.046	0.205	0.247	16.26	20.80	23.06
35	0.677	0.582	0.528	0.030	0.225	0.281	17.74	24.80	26.94
40	0.698	0.581	0.529	0.068	0.238	0.267	18.11	25.00	29.61
45	0.691	0.564	0.502	0.039	0.231	0.295	19.04	28.90	32.14
50	0.654	0.539	0.443	0.034	0.254	0.341	20.27	29.50	33.04
55	0.688	0.525	0.364	0.058	0.362	0.786	20.78	31.10	34.47
60	0.680	0.484	0.292	0.075	0.567	1.597	21.07	31.40	35.36
65	0.700	0.440	0.260	0.063	0.891	3.447	21.25	33.20	36.86

6.4.3 Network Size and Height

Experiments on different network sizes and heights were performed, and their effect on clustering accuracy (F-measure and SI) and speedup over centralized clustering were measured. Table 6.10 summarizes those results for the YAHOO dataset, and table 6.11 summarizes the same results for the SN dataset. The same results are illustrated in figure 6.9 and figure 6.10, respectively.

The first observation here is that for networks of height $H = 1$ ($\beta = 0$), the distributed clustering accuracy stays almost the same as the network size increases. This is evident through both the F-measure and SI. Since for networks of height 1 all nodes at level 0 are in the same neighborhood, every node can update its centroids based on complete information received from all other nodes at the end of each iteration (at the cost of increased communication). This means that increasing the network size does not affect

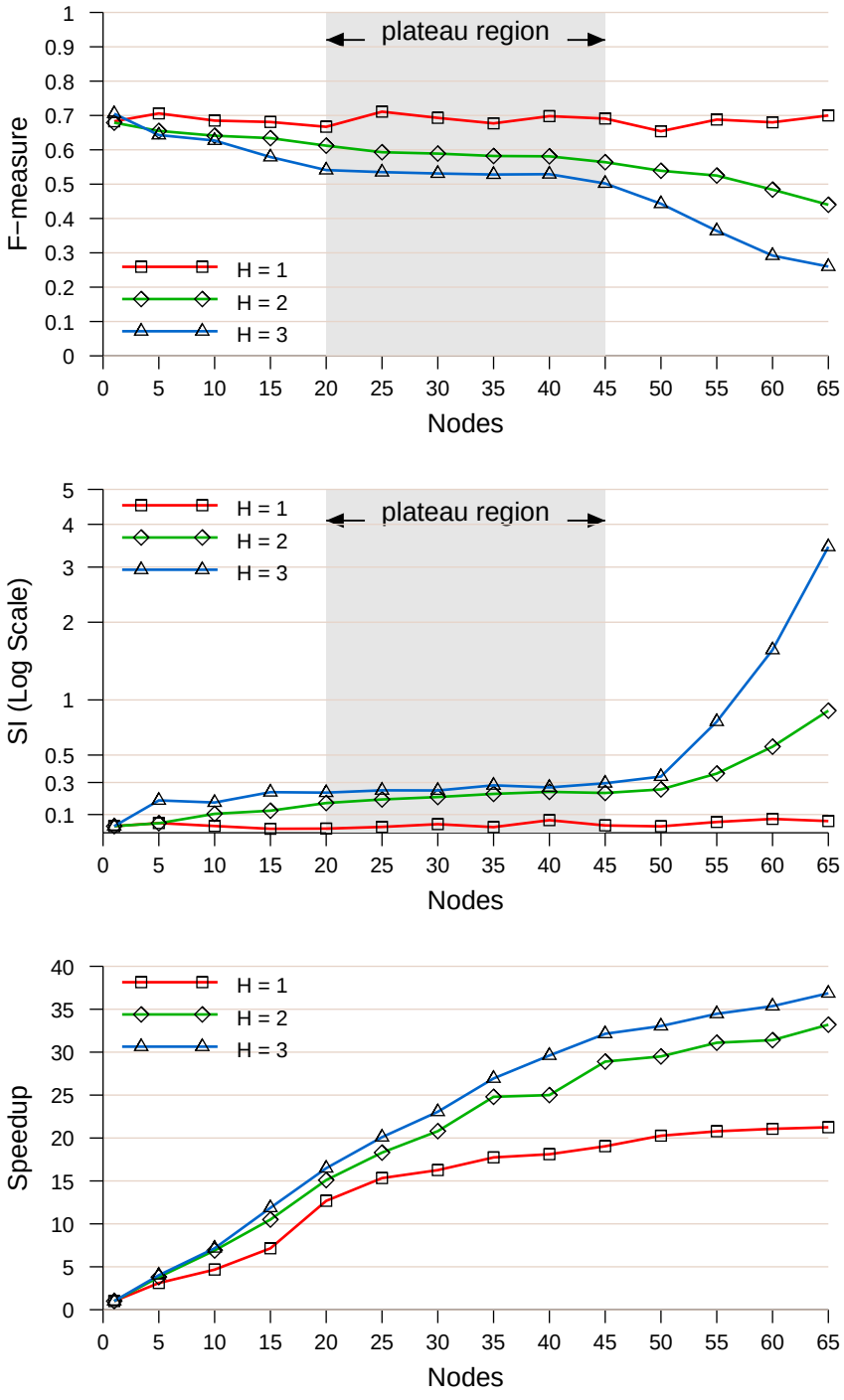


Figure 6.9: HP2PC Accuracy and Speedup [YAHOO]

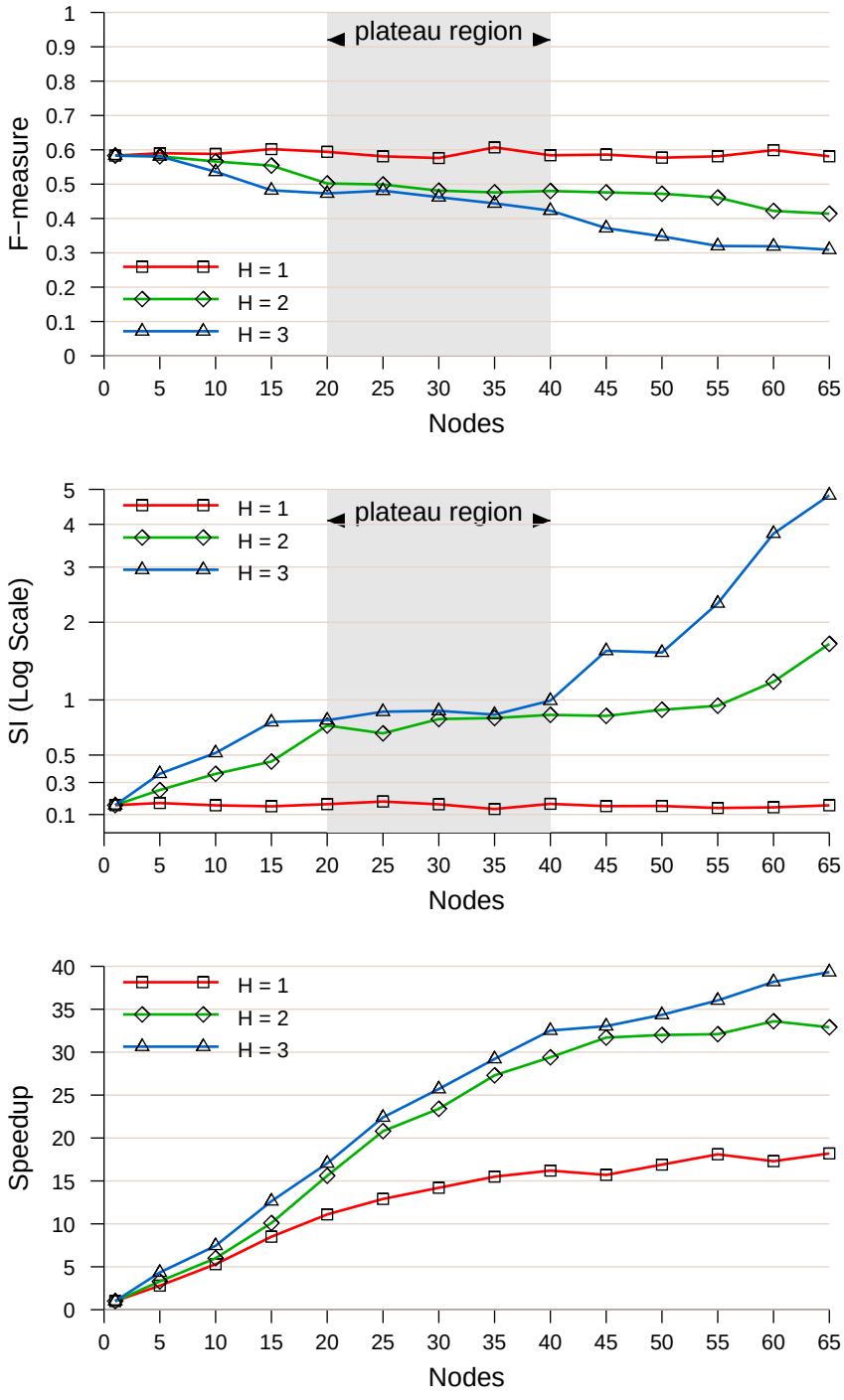


Figure 6.10: HP2PC Accuracy and Speedup [SN]

Table 6.11: Accuracy and Performance of HP2PC [SN]

	F-measure			SI			Speedup		
Nodes	H=1	H=2	H=3	H=1	H=2	H=3	H=1	H=2	H=3
1	0.581	0.592	0.602	0.155	0.135	0.144	1.00	1.00	1.00
5	0.590	0.581	0.581	0.168	0.253	0.361	2.80	3.30	4.34
10	0.588	0.566	0.536	0.154	0.367	0.517	5.30	6.00	7.45
15	0.602	0.554	0.482	0.148	0.451	0.782	8.50	10.10	12.64
20	0.594	0.502	0.473	0.161	0.752	0.799	11.10	15.60	17.05
25	0.581	0.499	0.481	0.177	0.680	0.881	12.90	20.80	22.40
30	0.576	0.481	0.462	0.160	0.811	0.889	14.20	23.40	25.71
35	0.607	0.476	0.444	0.132	0.828	0.852	15.50	27.30	29.21
40	0.584	0.480	0.423	0.163	0.857	0.993	16.20	29.40	32.52
45	0.586	0.476	0.372	0.149	0.843	1.584	15.70	31.70	33.03
50	0.577	0.472	0.348	0.150	0.904	1.561	16.90	32.00	34.35
55	0.581	0.461	0.320	0.138	0.942	2.311	18.10	32.10	36.03
60	0.599	0.422	0.319	0.142	1.202	3.764	17.30	33.60	38.19
65	0.581	0.414	0.309	0.154	1.680	4.819	18.20	32.90	39.33

accuracy of clustering, as long as it is of height 1.

The second observation is that, for networks of the same size, larger network heights cause clustering accuracy to drop. It is not surprising that this is the case, since at higher levels meta-clustering of lower level centroids is expected to produce some deviation from the true centroids. It is also noticeable that unlike networks of height 1, networks with height $H > 0$ tend to have less accuracy as the number of nodes is increased. As we keep H constant and increase N_P, the network partitioning factor, β, increases. This in turn means neighborhoods become smaller, thus causing the more accurate centroids at level 0 to become more fragmented.

An interesting observation is that there is a noticeable *plateau* region between the centralized case ($N_P = 1$), and a point where the data is finely partitioned ($N_P >$ some value), after which quality degrades rapidly. This plateau provides a clue on the relation

between the data set size and the number of nodes, beyond which the number of nodes should not be increased without increasing the data set size. An appropriate strategy for automatically detecting the higher boundary of this region (in scenarios where the network grows arbitrarily) is to compare the SI measure before and after adding nodes; if a sufficiently large difference in SI is noticed then network growth should be suspended until more data is available (and equally partitioned).

We investigate the effect of increasing hierarchy heights, as well as the accuracy at different levels within a single hierarchy, in more detail in the next subsections.

In terms of speedup, the trends show that the HP2PC algorithm exhibits decent speedup over the centralized case. For $H = 1$, however, speedup does not scale well with the network size, largely due to the increased communication cost for networks of that height. For $H > 0$, speedup becomes more scalable, as we can notice a big difference between $H = 1$ and $H = 2$ than between $H = 2$ and $H = 3$. This result carries an assertion that the hierarchical architecture of HP2PC is indeed scalable compared to flat P2P networks.

Comparison with P2P K-means

The accuracy of HP2PC is compared with P2P K-means [24], which is the current state-of-the-art in P2P-based distributed clustering. Since the implementation of P2P K-means is non-trivial, we used their benchmark synthetic dataset and results to compare against. The dataset is a two-dimensional mixture of 10 Gaussians, containing 78,200 points (referred hereto as `10G`). The actual data was not available from the authors, but rather the parameters of the Gaussians, which we used to re-generate the data[5]. The `10G` dataset is illustrated in figure 6.11.

[5]This means that there could be difference in the actual data points between our and their generated data due to the random number generation. However, we assume that the very large number of points will offset differences due to sampling

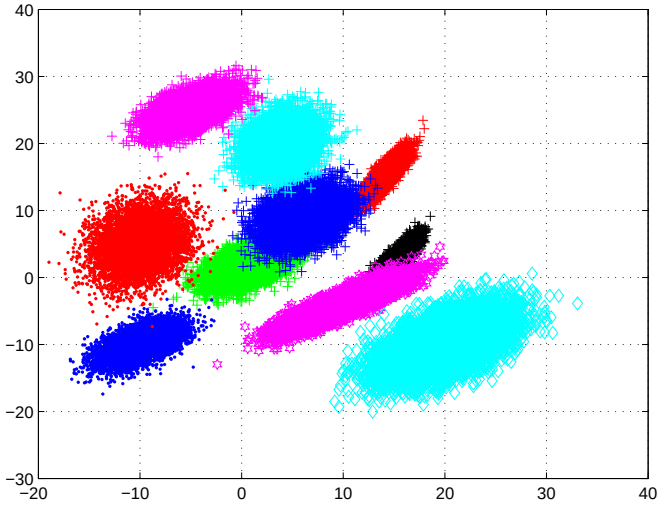


Figure 6.11: Two-dimensional Mixture of 10 Gaussians Dataset [10G]

The measure of accuracy in [24] was based on the difference between cluster membership produced by P2P K-means and that of the same data point as produced by the centralized K-means. To ensure accurate comparison, initial seeds for both the centralized and the P2P algorithms were the same. They report the total number of mislabeled data points as a percentage of the size of the dataset. The *percentage of mislabled points (PMP)* is

$$\frac{100|\{d \in D : L_{\text{cent}}(d) \neq L_{\text{p2p}}(d)\}|}{|D|}$$

Table 6.12 reports the results for P2P K-means and HP2PC (with various hierarchy heights). Nodes vary between 50 and 500, as reported in [24]. Figure 6.12 illustrates the trend in the results. HP2PC has zero error for networks of height 1, as expected. It is clear that for networks of low height, HP2PC is superior to P2P K-means. As the height increases, HP2PC starts to approach the error rate of P2P K-means ($H = 4, H = 5$), but

Table 6.12: PMP Comparison Between HP2PC and P2P K-means [10G]

Nodes	P2P K-Means	HP2PC				
		H=1	H=2	H=3	H=4	H=5
50	1.98	0.00	0.06	0.10	0.57	0.35
100	1.88	0.00	0.08	0.67	0.68	0.66
150	1.78	0.00	0.16	0.68	0.81	1.89
200	1.73	0.00	0.15	0.72	0.90	2.04
250	1.82	0.00	0.18	1.05	1.57	3.19
300	1.58	0.00	0.22	1.13	1.66	4.32
350	2.04	0.00	0.21	1.18	2.32	4.38
400	2.47	0.00	0.34	1.14	2.75	4.55
450	3.71	0.00	0.26	1.35	3.33	5.67
500	7.25	0.00	0.28	1.32	4.17	6.92

interestingly, HP2PC does not suffer from the sharp increase in PMP at very large number of nodes ($N_P > 300$).

P2P K-means has an advantage of being a model for unstructured P2P networks. It assumes that each node has a finite number of reachable neighbors, which are randomly selected from the nodes population. HP2PC, on the other hand, has a fixed hierarchical structure that allows it to produce superior results by avoiding random peering and propagation delay and error, a common disadvantage in P2P networks.

6.4.4 Clustering Quality at Different Hierarchy Levels

To test the effect of the hierarchical structure on clustering quality at different levels, we performed experiments on a network of size 250 nodes, and a fixed height of 5 ($\beta = 0.33$), on 20NG and RCV1. The number of nodes at each level are: 250, 83, 28, 9, 3, and 1, from level 0 to level 5, respectively. We compared the results to centralized k-means performed at each level of the hierarchy, and took the average over all neighborhoods on that level.

Figure 6.13 shows the accuracy achieved at each level of the hierarchy for 20NG, and

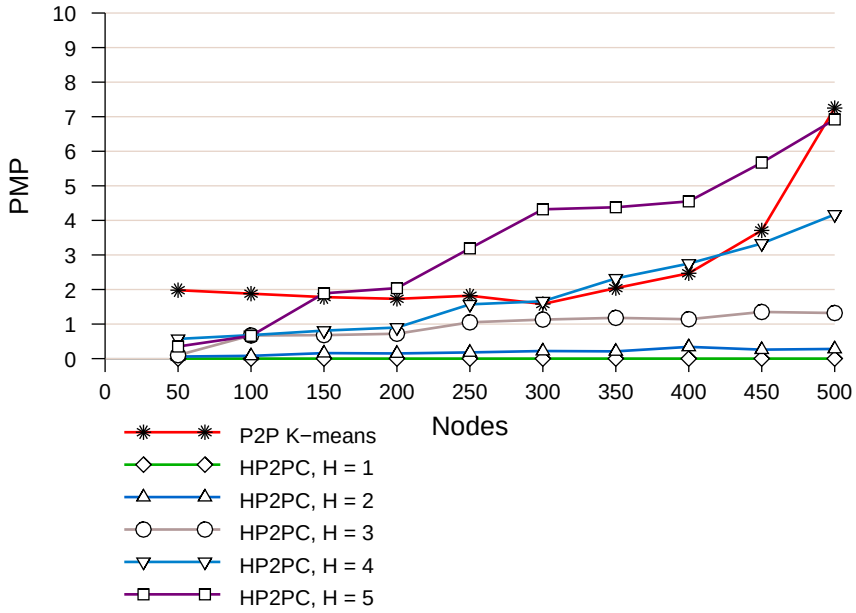


Figure 6.12: PMP Comparison Between HP2PC and P2P K-means [10G]

Table 6.13: Performance of HP2PC vs. Hierarchy Levels, $N_P = 250$, $\beta = 0.33$ [20NG]

			HP2PC		Centralized	
Level	N_P	N_Q	F-measure	SI	F-measure	SI
0	250	83	0.694	0.357	0.734	0.281
1	83	28	0.677	0.473	0.726	0.297
2	28	9	0.592	0.592	0.713	0.406
3	9	3	0.540	1.105	0.674	0.512
4	3	1	0.527	1.244	0.669	0.744
5	1	-	0.484	1.602	0.657	1.055

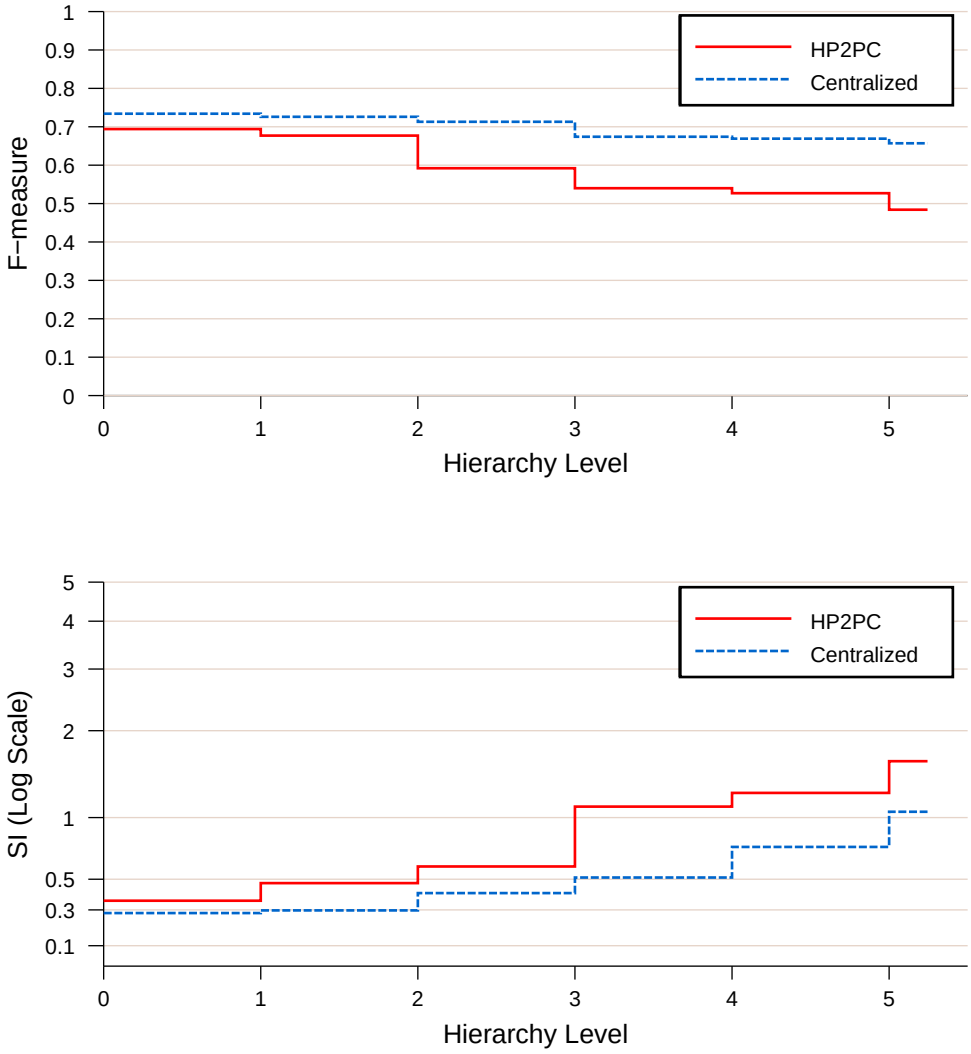


Figure 6.13: Clustering Accuracy vs. Hierarchy Level, $H = 5$ [20NG]

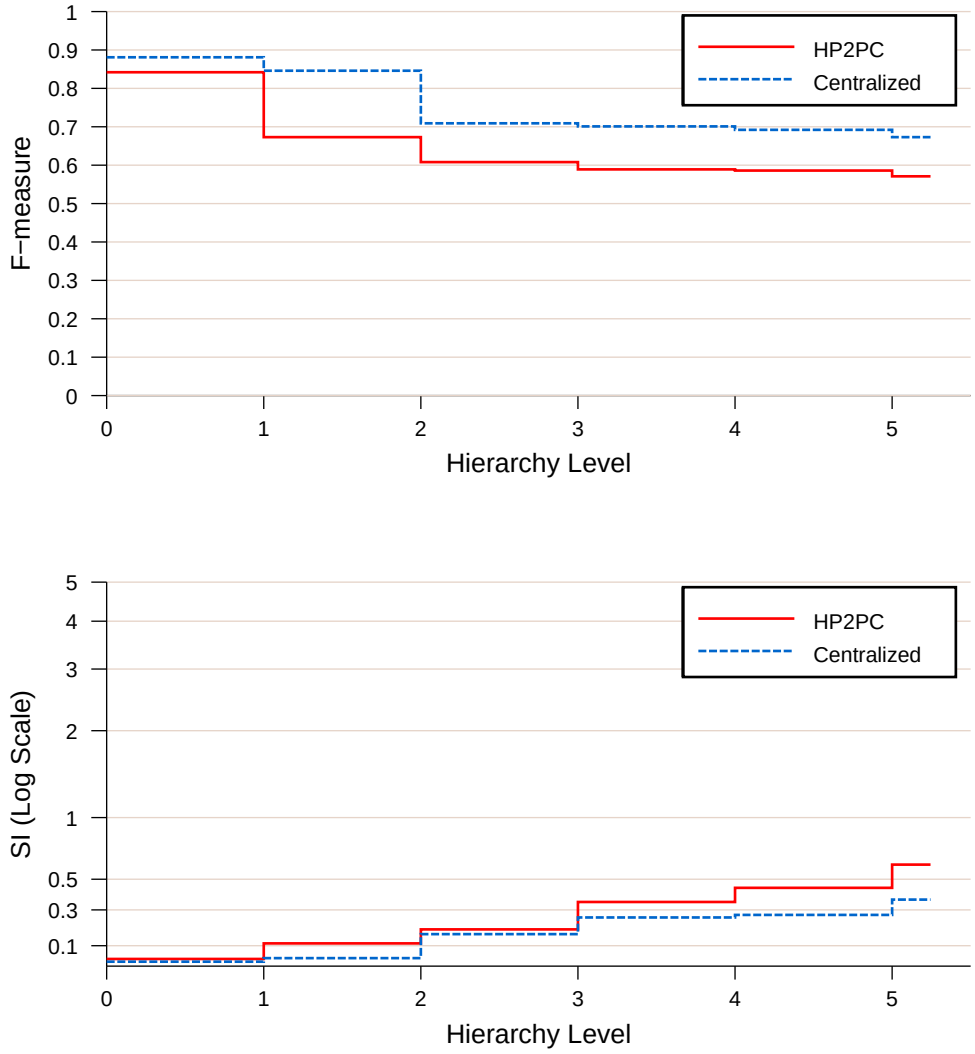


Figure 6.14: Clustering Accuracy vs. Hierarchy Level, $H = 5$ [RCV1]

Table 6.14: Performance of HP2PC vs. Hierarchy Levels $N_P = 250$, $\beta = 0.33$ [RCV1]

			HP2PC		Centralized	
Level	N_P	N_Q	F-measure	SI	F-measure	SI
0	250	83	0.842	0.034	0.881	0.021
1	83	28	0.673	0.112	0.846	0.038
2	28	9	0.608	0.187	0.709	0.161
3	9	3	0.589	0.349	0.701	0.255
4	3	1	0.586	0.441	0.692	0.270
5	1	-	0.571	0.606	0.673	0.364

compares it to the average centralized k-means accuracy at the same level. Figure 6.13(a) shows F-measure accuracy, while figure 6.13(b) shows SI change with hierarchy level. We notice that the clustering quality achieved by HP2PC is comparable to centralized k-means, and that it slightly degrades as we go up the hierarchy. Figure 6.14 shows the same results for RCV1, and verifies the same trend. Since at higher level of the hierarchy we only rely on cluster centroid information, this result is justifiable. Nevertheless, it is clear that at level 0 we can achieve clustering quality close to the centralized k-means algorithm. In scenarios where tall hierarchies are necessary (*e.g.* deep hierarchical organization) we can still achieve results that do not deviate much from the centralized case.

6.4.5 Hierarchy Height Scalability

Finally, we performed a set of experiments to test the effect of increasing hierarchy heights. Experimenting on tall hierarchies requires a large number of nodes so as to keep the neighborhood sizes reasonable. For this reason only the larger datasets 20NG and RCV1 were used in those experiments to avoid fine-grained partitioning of data across such a large number of nodes.

Table 6.15 reports the outcome of those experiments for a network of 250 nodes. The

Table 6.15: Performance of HP2PC vs. Hierarchy Heights, $N_P = 250$ [20NG, RCV1]

				20NG			RCV1		
Height	β	$N_Q^{(0)}$	$S_Q^{(0)}$	F-measure	SI	Speedup	F-measure	SI	Speedup
1	0.00	1	250.00	0.721	0.293	94.60	0.871	0.133	76.22
2	0.06	15	16.67	0.554	0.808	135.07	0.622	0.482	110.46
3	0.16	40	6.25	0.522	1.211	151.82	0.603	0.554	123.74
4	0.25	63	3.97	0.509	1.590	164.23	0.580	0.589	134.92
5	0.33	83	3.01	0.484	1.602	165.51	0.571	0.606	129.69
6	0.40	100	2.50	0.462	1.962	163.47	0.512	0.626	132.42
7	0.45	113	2.21	0.447	2.154	161.17	0.518	0.718	136.80
8	0.50	125	2.00	0.411	2.218	158.62	0.516	0.79	137.01
9	0.54	135	1.85	0.383	2.513	159.37	0.503	0.899	131.55
10	0.58	145	1.72	0.356	2.641	155.11	0.508	0.943	132.64

results are also reported in figure 6.15. We can see that as we increase the height clustering quality (which is measured at the root of the hierarchy) is affected (figure 6.15(a)). The sharpest decrease happens as soon as the height increases from 1 to 2, and then the degradation in F-measure and SI tend to stabilize. A similar trend can be seen in speedup (figure 6.15(c)). Both observations can be related to the size of neighborhoods at level 0 ($S_Q^{(0)}$), which decreases significantly when the height is increased from 1 (250 nodes) to 2 (16.67 nodes). From those observations we can conclude that neighborhood size plays a key role in determining both clustering quality and speedup. The more fine-grained neighborhoods in the network, the less the final clustering solution is dependent on the actual data (only available to level 0 nodes), because we have to go up the hierarchy several levels before we can converge to one solution for the whole network. Conversely, the more coarse-grained the neighborhoods, the better the final clustering solution is, due to creating more accurate clustering at lower levels before the less accurate merging of centroids takes place at higher levels of the hierarchy.

A similar argument can be made about speedup. The fewer nodes in a neighborhood,

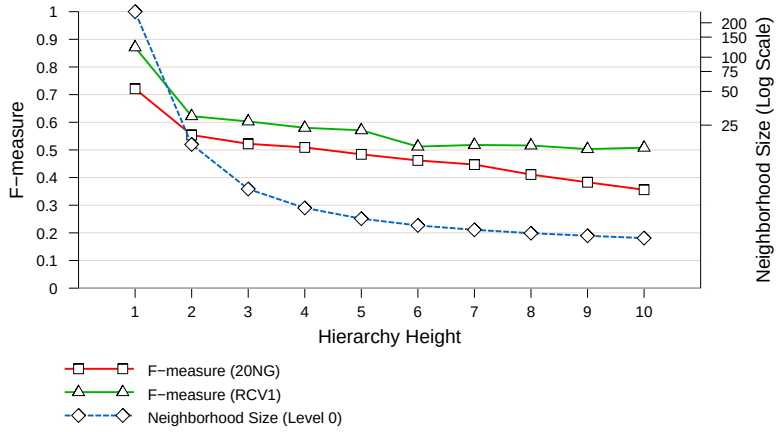
the less communication is needed between peers. However, from figure 6.15(c) we can see that we do not gain much speedup after a certain height (around $H = 4$ or 5). In fact speedup tend to decrease slightly after that point. This can be explained by looking at the size of neighborhoods in table 6.15. As soon as $S_Q^{(0)}$ decreases from 250 to 16.67 we notice a big jump in speedup (from 94.60 to 135.07). $S_Q^{(0)}$ then tends to decrease slowly as we increase the height, which after $H = 4$ stays almost the same. So in effect no gain is achieved; on the other hand, due to the increased height, we have to go through several cluster merging layers before the final solution is achieved. So our conclusion is that hierarchy height should not be increased unless there is a corresponding increase in the number of nodes at level 0.

6.4.6 Distributed Cluster Summarization

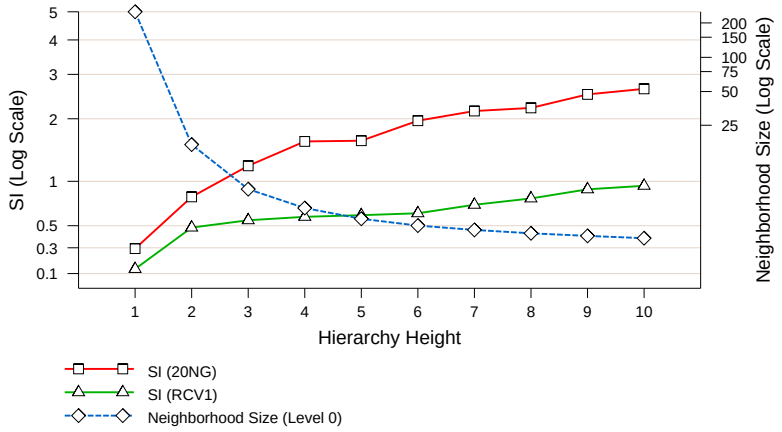
Generation of cluster summaries using the distributed version of CorePhrase was evaluated using different network sizes (N_p) and heights (H). Experiments were performed on the `20NG` dataset where the summary of each distributed cluster is compared to that of its centralized counterpart, and an average is taken over all clusters.

Figure 6.16 illustrates the accuracy of distributed cluster summarization compared to the baseline centralized cluster summarization. The first observation is that distributed cluster summaries can agree with their centralized counterparts with up to 88% ($H = 1, N_P = 50$) of the output keyphrases, which shows the feasibility of the distributed summarization algorithm.

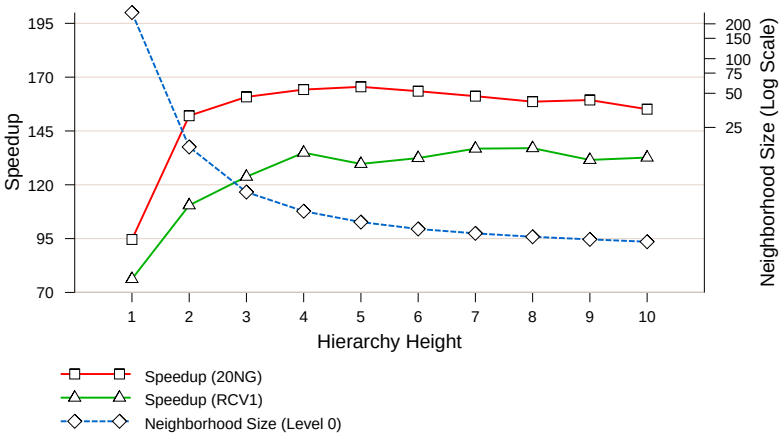
The second observation is that the number of top keyphrases, L, has a direct effect on accuracy. Lower values of L (usually lower than 100) tend to produce poor results, which is interpreted as being not enough keyphrases to accurately represent the core summaries exchanged between peers. Higher values (usually above 500) also may have negative effect.



(a) F-measure vs. Hierarchy Height



(b) SI vs. Hierarchy Height



(c) Speedup vs. Hierarchy Height

Figure 6.15: HP2PC Performance vs. Hierarchy Heights [`20NG, RCV1`]

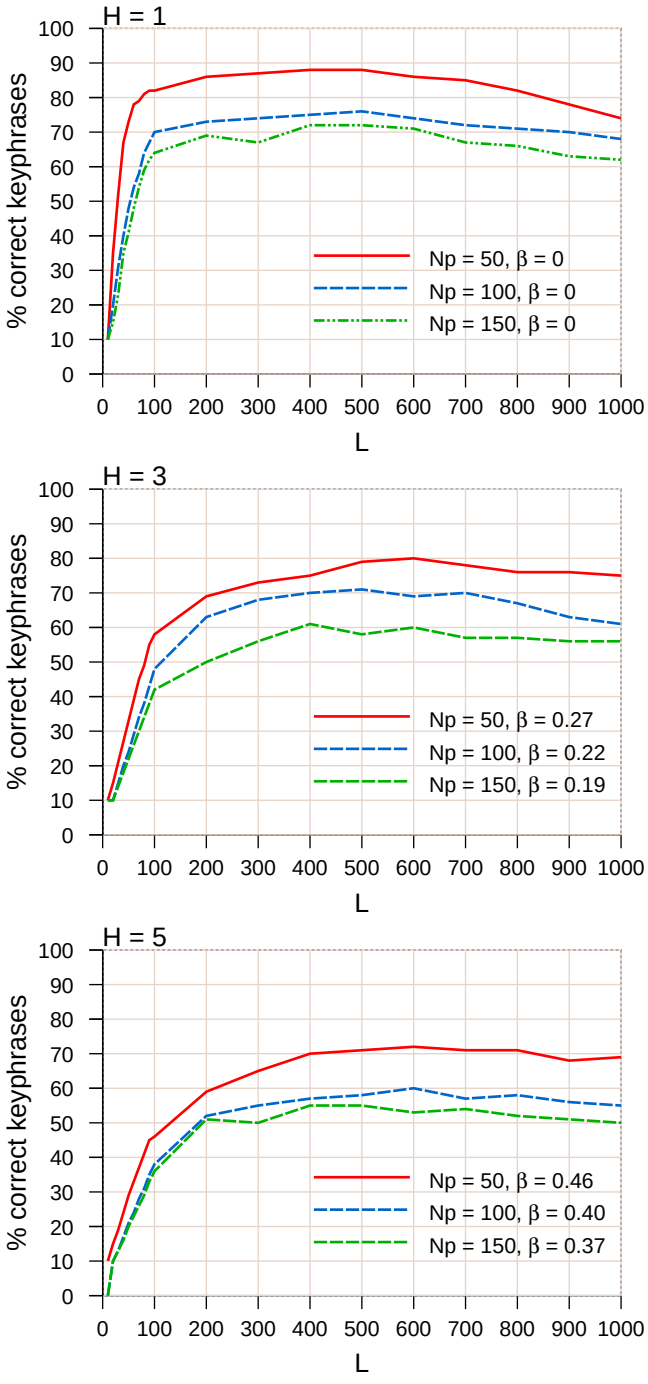


Figure 6.16: Distributed Cluster Summarization Accuracy [20NG]

For networks of low height (here $H = 1$), $100 < L < 500$ produces best results; while for those of larger heights (here $H = 3, 5$), $400 < L < 700$ produces best results. An interpretation of this observation is that accuracy of shorter hierarchies are less dependent on the less-accurate merging of keyphrases at higher levels. While on the the other hand, taller hierarchies require longer lists of core summaries to be carried over to higher levels to have enough keyphrases for merging, thus increasing the probability that intersection of summaries will not be empty.

The third observation is that networks of smaller number of nodes, N_P, produce more accurate results. Since the whole dataset is partitioned among N_P nodes, it is expected that a coarse-grained partitioning (smaller N_P) means that each node has access to larger portion of the distributed cluster, thus is able to get more accurate keyphrases.

To summarize those findings: (a) results of distributed cluster summarization can agree with centralized summarization with up to 88% accuracy ; (b) for networks of small height, $100 < L < 500$ should be used, while for networks of large height, $400 < L < 700$ should be used; and (c) accuracy of distributed summarization increases as the network size and height are decreased.

6.5 Summary

In this chapter an evaluation of the various methods and algorithms presented in this thesis was presented. Experiments were performed on actual document data sets representing different characteristics. Evaluation of the keyphrase extraction, collaborative distributed clustering, and the hierarchically-distributed clustering methods was presented and discussed.

Based on the experimental results, we can conclude that the distributed clustering

methods introduced in this thesis are successful with respect to their goal, with certain limitations that mostly can be accommodated. In addition to improving quality or gaining speedup, providing accurate interpretation for document clustering results through keyphrase extraction makes the results available for human interpretation. Detailed summary, conclusions, and recommendations are discussed in the next chapter.

CHAPTER 7

Conclusions and Future Research

In this thesis a set of methods and algorithms have been proposed to advance the performance of distributed document clustering. Distributed environments provide both opportunities and challenges for data mining. By building a collaborative scheme for clustering in distributed environments it was possible to construct locally-optimized clusters across a network of peer nodes, where collaboration between the nodes re-distributes data in such a way that clusters are finely tuned locally. We showed that collaboration between nodes can be limited if there is significant overlap between nodes in terms of their local data.

In the problem of globally-optimized distributed clustering, we proposed a hierarchically-distributed peer-to-peer architecture and a clustering algorithm that specifically addresses modularity and scalability of the network and consequently the scalability of the distributed clustering algorithm. We showed that good speedup can be achieved using this algorithm, with comparable results to centralized clustering.

7.1 Summary and Conclusions

Keyphrase Extraction

We presented the CorePhrase algorithm for accurately extracting descriptive keyphrases from text clusters, or multi-document sets in general. The algorithm is capable of discovering the topic of the cluster among the very few top extracted keyphrases. Usually the top two or three keyphrases are sufficient to cover the topic of the cluster; but cases where the very first keyphrase is sufficient are not uncommon. The algorithm is also domain-independent, requiring no prior knowledge about the data. The CorePhrase algorithm can be used in many applications; e.g. by any clustering algorithm for describing the discovered document clusters, or for creating a subject index of document collections.

Other uses of the algorithm include using the extracted keyphrases as representative features of the cluster, which could be employed to efficiently measure the similarity between clusters, or between a new document and a cluster, as used in the Collaborative P2P Clustering model.

Collaborative P2P Clustering

We have introduced a collaborative distributed approach for document clustering, which minimizes peer communication through cluster summarization. The major contribution of this work lies in three parts: (a) an incremental similarity-based clustering algorithm (b) an accurate document cluster summarization algorithm based on unsupervised keyphrase extraction, and (c) a collaborative document clustering algorithm based on exchanging cluster summaries, recommendation and aggregation of peer documents.

The results show significant improvement in the final clustering quality after aggregating

peer recommendations over the initial clustering solutions. The algorithm exhibits better improvements in the final clustering in networks with a larger number of nodes, by allowing nodes to expose more useful data to their peers, thus enhancing the global view of data of each node.

Hierarchically-Distributed P2P Clustering

We introduced a novel architecture and algorithm for distributed clustering, the Hierarchically-Distributed Peer-to-Peer Clustering model (HP2PC), which allows building hierarchical networks for clustering data. We demonstrated the flexibility of the model, showing that it achieves comparable quality to its centralized counterpart, and that it is possible to make it equivalent to traditional distributed clustering models (*e.g.* facilitator-worker models) by manipulating the neighborhood size and height parameters. We also demonstrated its superiority to a state-of-the-art algorithm.

7.1.1 Challenges

A number of challenges arose during this research, ranging from trivial to overwhelming. We list here some of those challenges and how they were addressed, so that others who follow this line of research become aware of what to expect.

- **Data**. Text mining is different than data mining. While data mining researchers enjoy a huge repository of standard machine learning datasets such as UCI [29], text mining has received little attention in this regard. Text mining researchers have access to a handful of standard datasets, most of which are prepared for information retrieval research, and some for natural language processing. The Reuters and 20-newsgroups datasets are probably the only datasets specifically created for text categorization,

and are used mainly in classification as opposed to clustering. We tried to adhere to those datasets, but also used other manually collected datasets, especially for keyphrase extraction, and made them available for others to use.

- **High dimensionality**. Again, unlike standard data mining research data which usually involve several dimensions, text data is of extremely high dimensionality (tens of thousands). To be able to properly handle such high dimensionality, a feature selection process is usually employed to bring the number of dimensions to a manageable level. Text feature selection is a research field on its own [7]. We used the simple Document Frequency feature selection method to reduce the number of features in large datasets. This reduced feature set was used during clustering. However, during keyphrase extraction the raw data was used to enable proper keyphrase identification.

- **Distributed computing** Simulation of peer-to-peer environments is not trivial. Initial attempts to use multiple machines for P2P experiments resulted in implementation and synchronization issues that were overwhelming. In addition, experiments that involves hundreds of nodes were not possible due to the limited number of computers available. Due to these limitation of resources, and in order to maintain simplicity and feasibility, we opted to do simulations on a single computer, rather than on a real network. We believe that proper P2P communication and synchronization in a real network should be delegated to an underlying middle-ware layer that provides an abstract communication model, similar to what MPI (Message Passing Interface) provides for traditional parallel computing. We encourage collaboration with researchers in distributed computing to facilitate such a P2P communication abstraction layer.

We also would like to point out some of the tricky issues in implementation and evaluation of the experiments.

- Keyphrase extraction works on raw text data, while clustering usually is done on vector space model data. Keeping two parallel data representations for each data set can be cumbersome, especially when automating experiments. Usually the output of clustering is cluster membership information (doc ID, cluster ID), which can be fed into the cluster summarizer, which in turn uses this information to summarize the raw document data from each cluster.

- The high-dimensional text data usually requires a large amount of memory to process, especially in raw format (as is the case in keyphrase extraction). The DIG model uses a few hundred megabytes for the medium collections (around 2000 documents). But fortunately this is not a big problem for two reasons. First, the memory usage levels off for larger collections since we only need to store the unique words in memory, along with some sentence path information. The number of unique words grows rapidly at the beginning (while processing documents incrementally), but later no more unique words are encountered. Second, we usually process a cluster of documents through DIG, not the entire collection. Clusters are usually of much smaller size than the entire document collection.

- In distributed clustering simulations, synchronization is usually not a problem since everything is done sequentially. Determining the sequence of operations in simulation is easy due to the simplicity of sequential processing. However, in real networks we may have race conditions and deadlocks, in addition to peculiarities of message passing protocols. A good simulator must be able to provide timing variations between nodes and implement those scenarios to mimic a real network.

- In implementation, it is recommended to separate the network handling logic (such as node formation, links, message passing, buffer management, etc.) from the clustering logic. Evaluation logic can also be separated into its own module, providing evaluation for both centralized and distributed clustering. This abstraction provides protection against network changes, and makes it easy to replace network implementation if the need arises.

- Two evaluation approaches can be chosen for distributed clustering, which can be complimentary. Evaluation against a reference categorization (ground truth), and evaluation against centralized clustering. When we evaluate against a reference categorization we are stating the absolute accuracy of the algorithm, which can be used to compare to any other clustering algorithm. In the absence of reference categorization we can only compare to the output of the centralized clustering algorithm, which gives us relative evaluation to that algorithm only.

7.1.2 Recommendations

An overview of the contributions in this thesis, and recommendations on when and where to use them, is illustrated in figure 7.1. Here we go through a few differentiators to decide on the applicability of certain algorithms suitable for each situation.

An initial differentiator is the nature of data distribution. In centralized environments, where the data is available in a central location, centralized clustering is a natural choice. For distributed data, we differentiate between two types of computing environments. On one hand we have parallel, grid, or cluster computing environments, where nodes are tightly coupled in a single cluster or supercomputer. In this situation parallel clustering, such as parallel k-means [27], is more appropriate. On the other hand we have P2P computing

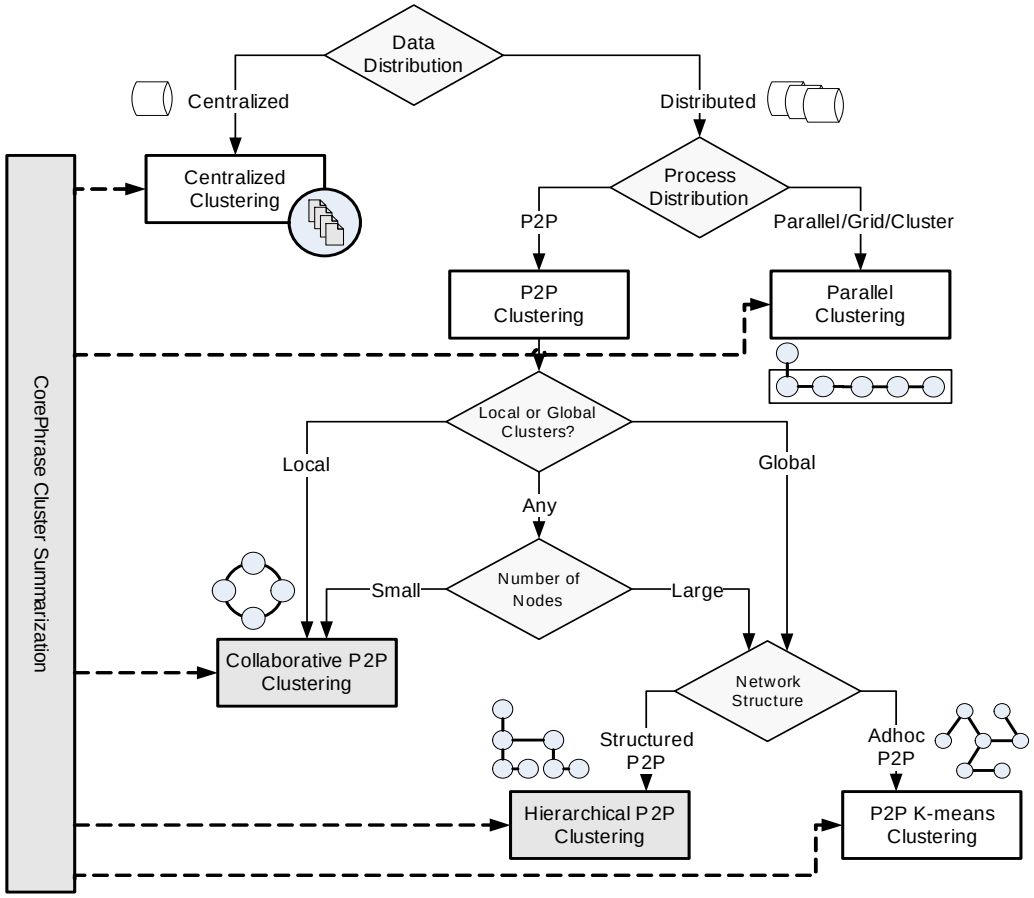


Figure 7.1: Contributions Flowchart

environments, where the nodes are usually loosely coupled.

In P2P environments we can differentiate the type of clusters required. For generating local clusters, the collaborative P2P clustering scheme is recommended for its ability to generate clusters around local data, while taking advantage of summarized cluster information from its peers. For global clusters we can make a choice between P2P K-means or HP2PC, depending on the network structure. For structured P2P networks, HP2PC is more appropriate due to its superiority in accuracy over P2P K-means. For ad-hoc P2P

networks, P2P K-means is a more appropriate choice since it can deal with arbitrary links between nodes.

If the type of distributed clusters (local or global) is not a requirement, we differentiate between small and large networks. For large networks, HP2PC is more appropriate due to its scalability. For smaller networks either collaborative P2P or HP2PC can be used.

7.2 Future Work

The field of distributed data mining (DDM) is still young, only about a decade old. With more advances in distributed computing, especially the emerging peer-to-peer and grid computing architectures, DDM will gain more advances as the underlying distributed architectures are further abstracted, providing a uniform and easy interface to use for higher level data mining tasks.

In the area of keyphrase extraction from clusters, future directions include using more features for the candidate phrases, based on heuristics of what constitutes a good phrase. Other ranking schemes are being investigated. Also the set of top keyphrases could be enhanced by removing spurious permutations and sub-phrases that appear in other phrases.

For distributed clustering in general, so far data partitioning has been done equally across nodes. Different partitioning strategies can be investigated to see the effect of unbalanced distribution of data in a network.

For collaborative clustering it would be interesting to let nodes make recommendations based on their *combined* view of all peer summaries, rather than one-to-one peer recommendation. Finally, it would be interesting in future work to let nodes summarize documents by doing single-document keyphrase extraction, then utilize such summarized documents to further enhance the clustering process through exchanging document summaries. Also

scalability can be further analyzed for this type of P2P clustering.

For the hierarchically-distributed clustering method we plan to extend this model to be dynamic, allowing nodes to join and leave the network, which requires maintaining a balanced network in terms of partitioning and height. We also plan to extend it to allow merging and splitting of complete hierarchies.

We are also investigating the possibility of improving the global clustering quality by by allowing centroids to cross neighborhoods through higher levels; *i.e.* clusters at lower level neighborhoods should be a function of higher level centroids. We believe that this will create an opportunity for better global clustering solutions, but at the expense of computational complexity.

Finally, the effect of HP2PC network structure and parameters on clustering accuracy also needs to be formally analyzed to provide error estimates that can guide the distributed clustering process. Error estimation can be done using sampling theory, which provides analysis tools for bounding the error [8]. This will also allow for sensitivity analysis against the various parameters, and may point out factors that need further evaluation.

7.3 List of Publications

The work in this thesis has resulted in a number of publications, as well as posters and software demos, which are listed below.

Book Chapters

- K. Hammouda and M. Kamel, "Data Mining in e-Learning", in Samuel Pierre (Ed.), E-Learning Networked Environments and Architectures: A Knowledge Processing Perspective, Springer, 2006.

Journal Articles – Published

- K. Hammouda and M. Kamel, "Distributed Collaborative Web Document Clustering Using Cluster Keyphrase Summaries", Information Fusion, Special Issue on Web Information Fusion. In Press, 2007.

Journal Articles – In Preparation

- K. Hammouda and M. Kamel, "Scalable Hierarchically-Distributed Peer-to-Peer Document Clustering", To be submitted to IEEE Transactions on Knowledge and Data Engineering.
- K. Hammouda and M. Kamel, "Simultaneous Document Clustering and Cluster Summarization in Peer-to-Peer Networks", To be submitted to IEEE Transactions on Knowledge and Data Engineering.

Conference Proceedings

- K. Hammouda and M. Kamel, "HP2PC: Scalable Hierarchically-Distributed Peer-to-Peer Clustering", 2007 SIAM Int. Conf. on Data Mining (SDM07), Minneapolis, MN, April 2007. In Press.
- Christopher Brooks, Scott Bateman, Wengang Liu, Gordon McCalla, Jim Greer, Dragan Gaevic, Timmy Eap, Griff Richards, Khaled Hammouda, Shady Shehata, Mohamed Kamel, Fakhri Karray, Jelena Jovanovic, "Issues and Directions with Educational Metadata", 3rd Annual Scientific Conference of the LORNET Research Network (I2LOR 2006), Montreal, QC, Nov 8-10, 2006.

- K. Hammouda and M. Kamel, "Collaborative Document Clustering", 2006 SIAM Conference on Data Mining (SDM06), pp. 453-463, Bethesda, Maryland, April 2006.

- K. Hammouda, D. Matute, and M. Kamel, "CorePhrase: Keyphrase Extraction for Document Clustering", Int. Conf. on Machine Learning and Data Mining (MLDM 2005), Springer LNAI 3587, pp. 265-274, Leipzig, Germany, July 2005.

Posters and Demos

- K. Hammouda and M. Kamel, "Automatic Metadata Extractor", *Demo* presented at the 2nd Annual Scientific Conference of the LORNET Research Network (I2LOR-05), Vancouver, BC, Nov 16-18, 2005. **First Prize Award for Best Demo**.

- K. Hammouda and M. Kamel, "Learning Object Similarity Ranking", *Demo* presented at the 2nd Annual Scientific Conference of the LORNET Research Network (I2LOR-05), Vancouver, BC, Nov 16-18, 2005.

- K. Hammouda and M. Kamel, "CorePhrase: Extracting Keyphrases from Documents", *Poster* presented at the 2nd Annual Scientific Conference of the LORNET Research Network (I2LOR-05), Vancouver, BC, Nov 16-18, 2005.

- K. Hammouda and M. Kamel, "Learning Object Content Summarization", *Poster & Demo* presented at the 3rd Annual Scientific Conference of the LORNET Research Network (I2LOR-06), Montreal, QC, Nov 8-10, 2006.

Bibliography

[1] K. Aas and L. Eikvil. Text categorisation: A survey. Technical Report 941, Norwegian Computing Center, June 1999.

[2] H. Ahonen, O. Heinonen, M. Klemettinen, and A. I. Verkamo. Applying data mining techniques for descriptive phrase extraction in digital document collections. In *Advances in Digital Libraries*, pages 2–11, 1998.

[3] H. Ahonen, O. Heinonen, M. Klemettinen, and A. I. Verkamo. Finding co-occurring text phrases by combining sequence and frequent set discovery. In R. Feldman, editor, *16th International Joint Conference on Articial Intelligence IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*, pages 1–9, 1999.

[4] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Company, 1999.

[5] J. Bakus, M.F. Hussin, and M. Kamel. Som-based document clustering using phrases. In *9th International Conference on Neural Information Processing (ICONIP'2002)*, pages 2212–2216, Singapore, November 2002.

[6] J. Bakus, M. Kamel, and T. Carey. Extraction of text phrases using hierarchical grammar. In *Advances in Artificial Intelligence: 15th Conference of the Canadian Society for Computational Studies of Intelligence (AI 2002)*, pages 319–324, Calgary, Canada, May 2002.

[7] Jan Bakus. *Feature selection and extraction for text classification.* PhD thesis, University of Waterloo, 2005.

[8] Sanghamitra Bandyopadhyay, Chris Giannella, Ujjwal Maulik, Hillol Kargupta, Kun Liu, and Souptik Datta. Clustering distributed data streams in peer-to-peer environments. *Information Sciences*, 176(14):1952–1985, 2006.

[9] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[10] D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[11] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the World Wide Web using WebACE. *AI Review*, 13(5-6):365–391, 1999.

[12] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27:329–341, 1999.

[13] Christopher Brooks, Scott Bateman, Wengang Liu, Gordon McCalla, Jim Greer, Dragan Gaevic, Timmy Eap, Griff Richards, Khaled Hammouda, Shady Shehata, Mohamed Kamel, Fakhri Karray, and Jelena Jovanovic. Issues and directions with

educational metadata. In *3rd Annual Scientific Conference of the LORNET Research Network (I2LOR 2006)*, Montreal, Canada, November 2006.

[14] Christopher Brooks, Mike Winters, Jim Greer, Gordon McCalla, James C. Lester, Rosa Maria Vicari, and Fabio Paraguau. The massive user modelling system (mums). In *International Conference on Intelligent Tutoring Systems (ITS 2004)*, volume 3220 of *LNCS*, pages 635–645. Springer, 2004.

[15] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann Publishers, 2003.

[16] K. Cios, W. Pedrycs, and R. Swiniarski. *Data Mining Methods for Knowledge Discovery*. Kluwer Academic Publishers, Boston, 1998.

[17] C. Clifton, R. Cooley, and J. Rennie. TopCat: data mining for topic identification in a text corpus. *IEEE Transactions on Knowledge and Data Engineering*, 16(8):949–964, August 2004.

[18] W. W. Cohen. Learning to classify English text with ILP methods. In *Proceedings of the 5^{th} International Workshop on Inductive Logic Programming*, pages 3–24. Department of Computer Science, Katholieke Universiteit Leuven, 1995.

[19] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: information and pattern discovery on the World Wide Web. In *Proceedings of the Ninth International Conference on Tools with Artifical Intelligence*, pages 558–567, 1997.

[20] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 509–516, Madison, US, 1998.

[21] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J.W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *16^{th} International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 126–135, 1993.

[22] J.C. da Silva, C. Giannella, R. Bhargava, H. Kargupta, and M. Klusch. Distributed data mining and agents. *Engineering Applications of Artificial Intelligence*, 18(7):791–807, 2005.

[23] Souptik Datta, Kanishka Bhaduri, Chris Giannella, Ran Wolff, and Hillol Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, 10(4):18–26, 2006.

[24] Souptik Datta, Chris Giannella, and Hillol Kargupta. K-means clustering over peer-to-peer networks. In *Workshop on High Performance and Distributed Mining (HPDM05). SIAM International Conference on Data Mining (SDM05)*, 2005.

[25] Souptik Datta, Chris Giannella, and Hillol Kargupta. K-means clustering over a large, dynamic network. In *SIAM International Conference on Data Mining (SDM06)*, pages 153–164, 2006.

[26] J. A. Delgado. *Agent-Based Information Filtering and Recommender System On The Internet*. PhD thesis, Dept. of Intelligence Computer Science, Nagoya Institute of Technology, 2000.

[27] Inderjit S. Dhillon and Dharmendra S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining*, volume 1759 of *LNAI*, pages 245–260. Springer, 2000.

[28] Mark d'Inverno and Michael Luck. *Understanding Agent Systems*. Springer, 2001.

[29] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[30] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7^{th} International Conference on Information and Knowledge Management*, pages 148–15, November 1998.

[31] J. C. Dunn. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetica*, 4:95–104, 1974.

[32] M. Eisenhardt, W. Muller, and A. Henrich. Classifying documents by distributed p2p clustering. In *Informatik 2003: Innovative Information Technology Uses*, Frankfurt, Germany, 2003.

[33] Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, Portland, Oregon, 1996. AAAI Press.

[34] U. Fayyad and R. Uthurusamy. Data mining and knowledge discovery in databases: Introduction to the special issue. *Communications of the ACM*, 39(11), November 1999.

[35] R. Feldman and I. Dagan. Knowledge discovery in textual databases (kdt). In *First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pages 112–117, Montreal, Canada, 1995.

[36] R. Feldman, M. Fresko, Y. Kinar, Y. Lindell, O. Liphstat, M. Rajman, Y. Schler, and O. Zamir. Text mining at the term level. In *Principles of Data Mining and Knowledge Discovery*, pages 65–73, 1998.

[37] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In *IJCAI*, pages 668–673, 1999.

[38] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Somers, and R. Evans. Software agents: A review. Technical Report TCD-CS-1997-06, Trinity College, University of Dublin, 1997.

[39] K. Hammouda. Web mining: Identifying document structure for web document clustering. MASc thesis, Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada, 2002.

[40] K. Hammouda and M. Kamel. Phrase-based document similarity based on an index graph model. In *2002 IEEE International Conference on Data Mining (ICDM02)*, pages 203–210, Maebashi, Japan, December 2002. IEEE Computer Society.

[41] K. Hammouda and M. Kamel. Incremental document clustering using cluster similarity histograms. In *The 2003 IEEE/WIC International Conference on Web Intelligence (WI03)*, pages 597–601, Halifax, Canada, October 2003.

[42] K. Hammouda and M. Kamel. Document similarity using a phrase indexing graph model. *Knowledge and Information Systems*, 6(6):710–727, November 2004.

[43] K. Hammouda and M. Kamel. Efficient phrase-based document indexing for web document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1279–1296, October 2004.

[44] K. Hammouda and M. Kamel. Corephrase: Keyphrase extraction for document clustering. In P. Perner and A. Imiya, editors, *IAPR International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM 2005)*, volume 3587 of *LNAI*, pages 265–274, Leipzig, Germany, July 2005. Springer.

[45] K. Hammouda and M. Kamel. Collaborative document clustering. In *Proceedings of the Sixth SIAM International Conference on Data Mining (SDM06)*, pages 453–463, Bethesda, MD, April 2006.

[46] K. Hammouda and M. Kamel. Data mining in e-learning. In Samuel Pierre, editor, *E-Learning Networked Environments and Architectures: A Knowledge Processing Perspective*, pages 374–404. Springer, 2006.

[47] K. Hammouda and M. Kamel. Distributed collaborative web document clustering using cluster keyphrase summaries. *Information Fusion, Special Issue on Web Information Fusion*, 2007. In Press.

[48] K. Hammouda and M. Kamel. HP2PC: Scalable hierarchically-distributed peer-to-peer clustering. In *Seventh SIAM International Conference on Data Mining (SDM07)*, April 2007. Accepted.

[49] Jiawei Han and Micheline Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2nd edition, 2006.

[50] M. Hatala, G. Richards, T. Eap, and J. Willms. The edusource communication language: Implementing an open network for learning object repositories and services. In *Proceedings of the ACM Symposium on Applied Computing (SAC) 2004, Special Track on Engineering e-Learning Systems*, pages 957–962, Nicosia, Cyprus, 2004.

[51] T. H. Haveliwala, A. Gionis, and P. Indyk. Scalable techniques for clustering the web. *WebDB (Informal Proceedings)*, pages 129–134, 2000.

[52] M. A. Hearst. Untangling text data mining. In *ACL'99: the 37th Annual Meeting of the Association for Computational Linguistics*, pages 3–10, 1999.

[53] T. Hofmann. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence IJCAI-99*, pages 682–687, 1999.

[54] S. J. Hong and S. M. Weiss. Advances in predictive model generation for data mining. Technical Report RC-21570, IBM Research, 1999.

[55] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. Newsgroup exploration with websom method and browsing interface. Technical Report A32, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, January 1996.

[56] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. WEBSOM—self-organizing maps of document collections. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps*, pages 310–315, Espoo, Finland, June 1997.

[57] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, N.J., 1988.

[58] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[59] J. Jang, C. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing — A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, 1997.

[60] Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle. DBDC: Density based distributed clustering. In *EDBT*, pages 88–105, 2004.

[61] Z. Jiang, A. Joshi, R. Krishnapuram, and L. Yi. Retriever: Improving web search engine results using clustering. Technical report, CSEE Department, UMBC, 2000.

[62] Erik L. Johnson and Hillol Kargupta. Collective, hierarchical clustering from distributed, heterogeneous data. In *Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD*, pages 221–244. Springer-Verlag, 2000.

[63] Hillol Kargupta, Ilker Hamzaoglu, and Brian Stafford. Scalable, distributed data mining using an agent-based architecture. In *Proceedings of Knowledge Discovery and Data Mining*, pages 211–214. AAAI Press, 1997.

[64] Hillol Kargupta, Weiyun Huang, Krishnamoorthy Sivakumar, and Erik Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.

[65] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen. Creating an order in digital libraries with self-organizing maps. In *Proceedings of WCNN'96, World Congress on Neural Networks*, pages 814–817, September 1996.

[66] J. Kay, N. Masionneuve, K. Yacef, and O. Zaiane. Determination of factors influencing the achievement of the first-year university students using data mining methods. In *Proceedings of the Workshop on Educational Data Mining at the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)*, 45–52.

[67] Matthias Klusch, Stefano Lodi, and Gianluca Moro. Agent-based distributed data mining: The KDEC scheme. In *AgentLink*, pages 104–122, 2003.

[68] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, 1995.

[69] R. Kosala and H. Blockeel. Web mining research: a survey. *ACM SIGKDD Explorations Newsletter*, 2(1):1–15, 2000.

[70] R. Krishnapuram, A. Joshi, and L. Yi. A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering. In *IEEE International Conference on Fuzzy Systems*, pages 1281–1286, Korea, August 1999.

[71] S. Krishnaswamy, S. W. Loke, and A. Zaslavsky. Cost models for heterogeneous distributed data mining. In *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 31–38, Chicago, IL, 2000.

[72] Anup Kumar, Mehmed Kantardzic, and Samuel Madden. Guest editors' introduction: Distributed data mining–framework and implementations. *IEEE Internet Computing*, 10(4):15–17, 2006.

[73] K. Lagus, T. Honkela, S. Kaski, and T. Kohonen. Self-organizing maps of document collections: A new approach to interactive exploration. In Evangelios Simoudis, Jiawei Han, and Usama Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 238–243. AAAI Press, Menlo Park, California, 1996.

[74] B. Lent, R. Agrawal, and R. Srikant. Discovering trends in text databases. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 227–230. AAAI Press, 1997.

[75] D. D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[76] J. Li and R. Morris. Document clustering for distributed fulltext search. In *2nd MIT Student Oxygen Workshop*, Cambridge, MA, August 2002.

[77] Marty Lucasv. Mining in textual mountains: An interview with Marti Hearst. Mappa Mundi, http://mappa.mundi.net/trip-m/hearst/, 1999.

[78] M.J. Mana-Lopez, M. De Buenaga, and J.M. Gómez-Hidalgo. Multidocument summarization: An added value to clustering in interactive retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):215–241, April 2004.

[79] I. Mani and E. Bloedorn. Multi-document summarization by graph search and matching. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-97)*, pages 622–628, Providence, RI, 1997.

[80] Agathe Merceron and Kalina Yacef. Educational data mining: A case study. In *The 12th International Conference Artificial Intelligence in Education (AIED 2005)*, pages 467–474, Amsterdam, July 2005.

[81] Agathe Merceron and Kalina Yacef. TADA-Ed for educational data mining. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 7(1), May 2005.

[82] D. Merkl and A. Rauber. Automatic labeling of self-organizing maps for information retrieval. In *Proceedings of the 6th International Conference on Neural Information Processing (ICONIP99)*, pages 37–42, Perth, Australia, November 1999.

[83] Srujana Merugu and Joydeep Ghosh. Privacy-preserving distributed clustering using generative models. In *IEEE International Conference on Data Mining (ICDM)*, pages 211–218, 2003.

[84] T. Michell. *Machine Learning*. McGraw Hill, 1997.

[85] David Monk. Using data mining for e-learning decision making. *The Electronic Journal of e-Learning*, 3(1):41–54, 2005.

[86] Martin Muehlenbrock. Automatic action analysis in an interactive learning environment. In *Workshop on Usage Analysis in Learning Systems, The 12th International Conference on Artificial Intelligence in Education (AIED 2005)*, pages 73–80, Amsterdam, Netherlands, July 2005.

[87] U. Y. Nahm and R. J. Mooney. A mutually beneficial integration of data mining and information extraction. In *17th National Conference on Artificial Intelligence (AAAI-00)*, pages 627–632, 2000.

[88] J. Neto, A. Santos, C. Kaestner, and A. Freitas. Document clustering and text summarization. In *Proc. 4th International Conference Practical Applications of Knowledge Discovery and Data Mining (PADD-2000)*, pages 41–55, London, UK, January 2000.

[89] R. Osdin, I. Ounis, and R.W. White. Using hierarchical clustering and summarisation approaches for web retrieval: Glasgow at the TREC 2002 interactive track. In *The 11th Text REtrieval Conference (TREC 2002)*, pages 640–644, Maryland, USA, November 2002.

[90] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[91] D. Radev and W. Fan. Automatic summarization of search engine hit lists. In *ACL'2000 Workshop on Recent Advances in Natural Language Processing and Information Retrieval*, pages 99–109, Hong Kong, October 2000.

[92] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, November 1975.

[93] Nagiza F. Samatova, George Ostrouchov, Al Geist, and Anatoli V. Melechko. RACHET: An efficient cover-based merging of clustering hierarchies from distributed datasets. *Distributed and Parallel Databases*, 11(2):157–180, 2002.

[94] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD-2000 Workshop on TextMining*, August 2000.

[95] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64, Austin, TX, July 2000. AAAI.

[96] Alexander Strehl. *Relationship-based Clustering and Cluster Ensembles for High-Dimensional Data Mining*. PhD thesis, Faculty of Graduate School, The University of Texas at Austin, 2002.

[97] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, December 2002.

[98] C. Y. Suen. N-gram statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):164–172, April 1979.

[99] J.F. Superby, J-P. Vandamme, and N. Meskens. Determination of factors influencing the achievement of the first-year university students using data mining methods. In *Proceedings of the Workshop on Educational Data Mining at the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)*, 37–44.

[100] A.-H. Tan. Text mining: The state of the art and the challenges. In *Pacific Asia Conference on Knowledge Discovery and Data Mining PAKDD'99 workshop on Knowledge Discovery from Advanced Databases*, pages 65–70, 1999.

[101] Tom Payne Titus Winters, Christian Shelton and Guobiao Mei. Topic extraction from item-level grades. In *Workshop on Educational Data Mining, 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 7–14, Pittsburgh, PA, 2005.

[102] Peter D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.

[103] R. Weiss, B. Velez, M. A. Sheldon, C. Namprempre, P. Szilagyi, A. Duda, and D. K. Gifford. Hypursuit: a hierarchical network search engine that exploits content-link hypertext clustering. In *Hypertext'96: The 7th ACM Conference on Hypertext*, pages 180–193, 1996.

[104] S. M. Weiss, C. Apté, F. Damerau, D. E. Johnson, F. J. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69, 1999.

[105] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[106] Ran Wolff, Kanishka Bhaduri, and Hillol Kargupta. Local L2-thresholding based data mining in peer-to-peer systems. In *SIAM International Conference on Data Mining (SDM06)*, pages 430–441, 2006.

[107] W. Wong and A. Fu. Incremental document clustering for web page classification. In *2000 International Conference on Information Society in the 21th Century: Emerging Technologies and New challenges (IS2000)*, Japan, 2000.

[108] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

[109] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of the 21st Annual International ACM SIGIR Conference*, pages 46–54, Melbourne, Australia, 1998.

[110] O. Zamir, O. Etzioni, O. Madanim, and R. M. Karp. Fast and intuitive clustering of web documents. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 287–290, Newport Beach, CA, August 1997. AAAI.